Уральский федеральный университет
имени первого Президента России Б.Н.Ельцина
Институт новых материалов и технологий

С. К. БУЙНАЧЕВ
Е. Е. БАЖЕНОВ
И. В. ТРОИЦКИЙ

МОДЕЛИРОВАНИЕ ДВИЖЕНИЯ И НАГРУЗОК ПЛОСКИХ МЕХАНИЗМОВ НА ЯЗЫКЕ PYTHON

Учебное пособие

Министерство науки и высшего образования
Российской Федерации

Уральский федеральный университет
Имени первого Президента России Б. Н. Ельцина

С. К. Буйначев
Е. Е. Баженов
И. В. Троицкий

МОДЕЛИРОВАНИЕ ДВИЖЕНИЯ И НАГРУЗОК ПЛОСКИХ МЕХАНИЗМОВ НА ЯЗЫКЕ PYTHON

Учебное пособие

Рекомендовано методическим советом
Уральского федерального университета
для студентов вуза, обучающихся
по направлениям подготовки
15.03.02, 15.04.02 «Технологические
машины и оборудование»

Екатеринбург
Издательство Уральского университета
2019

УДК 621.837.3:004.432Phyton(075.8)
ББК 34.412.5я73+32.973.22я73
Б90

Рецензенты:
Г. О. Котиев, д-р техн. наук, проф., завкафедрой «Колесные машины» Московского государственного технического университета им. Н. Э. Баумана;
И. П. Трояновская, д-р техн. наук, доц. кафедры «Теоретическая механика и теория механизмов и машин», проф. кафедры «Прикладная механика» (ФГБОУ ВО Южно-Уральский государственный аграрный университет)

Научный редактор — канд. техн. наук, доц. *С. В. Бутаков*

Буйначев, С. К.
Б90 Моделирование движения и нагрузок плоских механизмов на языке Python : учебное пособие / С. К. Буйначев, Е. Е. Баженов, И. В. Троицкий. — Екатеринбург : Изд-во Урал. ун-та, 2019. — 80 с.
ISBN 978-5-7996-2685-3

Учебное пособие соответствует программам курсов «Теория механизмов и машин», «Математическое моделирование» и включает необходимые сведения для их изучения. Пособие содержит краткие описания программ на языке Python.

Может быть рекомендовано студентам технических вузов, занимающихся программированием, а также служить справочным материалом для выполнения проектных работ, связанных с выполнением на компьютере.

Библиогр.: 5 назв. Рис. 19.

УДК 621.837.3:004.432Phyton(075.8)
ББК 34.412.5я73+32.973.22я73

ISBN 978-5-7996-2685-3

Предисловие

Развитие вычислительной техники и массовое внедрение компьютеров в исследование механизмов повысили требования к математической подготовке инженеров. В настоящее время процесс проектирования машины требует использования алгоритмического языка высокого уровня.

Самым удобным признан язык Python. Многие конструкции языков сходны, однако на Python программирование имеет существенные преимущества. Этот язык выбран не случайно, поскольку разработанные авторские программы, представленные в пособии, работают без изменения под разными операционными системами Windows, Linux, MacOS.

В учебном пособии представлен объективный подход к описанию групп Ассура и их поведению совместно с электродвигателем с дальнейшим анализом и визуализаций решений.

Пособие составлено как набор модулей для разных задач, позволяющих быстро найти нужную информацию и освоить новые приемы программирования.

1. Плоские механизмы. Общие сведения

Механизм — часть машины, в которой рабочий процесс реализуется путем выполнения определенных механических движений. Чаще всего эти движения являются плоскими. Однотипные механизмы используются в конструкциях самых разных по назначению и условиям работы машин.

В плоском механизме положение звена можно описать тремя координатами — двумя линейными и одной угловой [1]. Из множества кинематических цепей с абсолютно жесткими звеньями рассмотрим только цепи со степенью подвижности $w = 1$. Для цепи с координатами звеньев

$$\vartheta = (\vartheta_1,.., \vartheta_n)^{T,} \quad (1)$$

где ϑ_i — i-я координата j-го звена,

n — число координат звеньев кинематической цепи.

Зададим ведущему звену закон движения $\varphi(t)$, это позволит определить координаты всех ведомых звеньев:

$$\vartheta = \vartheta(\varphi(t)). \quad (2)$$

Если известна скорость ведущего звена $\varphi'(t)$ и его ускорение $\varphi'(t)$, то известны скорость ϑ' и ускорение ϑ'' ведомых звеньев

$$\vartheta' = \vartheta_\varphi \varphi', \quad (3)$$

$$\vartheta'' = \vartheta_\varphi \varphi'' + \vartheta_{\varphi\varphi} (\varphi')^2, \quad (4)$$

где

$$\varphi' = \partial\varphi/\partial t, \tag{5}$$

$$\varphi'' = \partial^2\varphi/\partial t^2. \tag{6}$$

Частные производные по координатам являются аналогами скоростей и ускорений по координате ведущего звена.

$$\vartheta_\varphi = (\partial\vartheta_1/\partial\varphi,..,\partial\vartheta_n/\partial\varphi)^T, \tag{7}$$

$$\vartheta_{\varphi\varphi} = (\partial^2\vartheta_1/\partial\varphi^2, ...,\partial\vartheta^2{}_n/\partial\varphi^2)^T. \tag{8}$$

Кинетическая энергия звеньев представлена квадратичной формой

$$T = \frac{1}{2}\ \vartheta'^T\Theta\ \vartheta'. \tag{9}$$

Если форма каноническая, то матрица коэффициентов инерции звеньев

$$\Theta = \mathrm{diag}\{\Theta_i\}_{\ i=1,\,n}. \tag{10}$$

Подставим в выражение кинетической энергии (9) в формулу (3)

$$T = \frac{1}{2}\ (\vartheta_\varphi\varphi')^T\Theta\vartheta_\varphi\varphi' = \frac{1}{2}\ \vartheta_\varphi{}^T\Theta\vartheta_\varphi\varphi'^2 = \frac{1}{2}\ J\varphi'^2, \tag{11}$$

где J — приведенный момент инерции.

Обозначим приведенный момент инерции звеньев к ведущей координате

$$J = \vartheta_\varphi{}^T\Theta\ \vartheta_\varphi \tag{12}$$

и его производную по обобщенной координате

$$J_\varphi = 2\vartheta_\varphi{}^T\Theta\ \vartheta_{\varphi\varphi}. \tag{13}$$

В этом случае задача сводится к исследованию вращения маховика с переменным приведенным моментом инерции. Он определяется конфигурацией рычажной системы [2]

$$J = J(\varphi). \tag{14}$$

Свойства приведенного момента характеризуются гладкой и неотрицательной функцией:

$$J(\varphi) = \Sigma_{\iota=1,\, n} \Theta_i (\vartheta_{i\varphi})^2 \geqslant 0, \tag{15}$$

в которой период изменения соответствует обороту кривошипа

$$J(\varphi) = J(\varphi + 2\pi). \tag{16}$$

Пусть на кинематическую цепь по координатам χ действует вектор внешних сил

$$X = X(\chi(\varphi)). \tag{17}$$

Работа вектора X по координатам χ есть скалярное произведение:

$$A = \chi^T X. \tag{18}$$

Уравнение движения найдем как экстремаль функционала [3]:

$$\int_{(0,\, t)} (A + T)\, dt \to \min_{\varphi,\, \chi}. \tag{19}$$

Координаты φ и χ связаны с обобщенными координатами q уравнениями

$$\tau(q, \vartheta) = 0, \tag{20}$$

$$\alpha(q, \chi) = 0. \tag{21}$$

Добавим в функционал неопределенные множители Λ_T и Λ_E

$$\dim\langle\Lambda_T\rangle = \dim\langle\varphi\rangle, \tag{22}$$

$$\dim\langle\Lambda_E\rangle = \dim\langle\chi\rangle. \tag{23}$$

Теперь функционал

$$\int_{(0,\, t)} (A + T - \tau'\Lambda_T - \alpha'\Lambda_E)\, dt \to \min_{\varphi,\, \chi q} \tag{24}$$

имеет единственное решение.

Уравнения Лагранжа первого рода

$$J\varphi'' + \frac{1}{2} J_\varphi \varphi'^2 + \tau_\varphi' \Lambda_T = 0, \tag{25}$$

$$-X + \alpha_\chi' \Lambda_E = 0, \tag{26}$$

$$\tau_q' \Lambda_T + \alpha_q' \Lambda_E = 0. \tag{27}$$

Получим движение одномассовой кинематической цепи, описанной уравнением

$$J\varphi'' + \frac{1}{2} J_\varphi \varphi'^2 = M. \tag{28}$$

Преобразуем уравнение движения:

$$\vartheta_\varphi^T \Theta \vartheta_\varphi \varphi'' + \vartheta_\varphi^T \Theta \vartheta_{\varphi\varphi} \varphi'^2 = \chi_\varphi^T X, \tag{29}$$

и еще раз

$$\vartheta_\varphi^T \Theta (\vartheta_\varphi \varphi'' + \vartheta_{\varphi\varphi} (\varphi')^2) = (\chi_\vartheta \vartheta_\varphi)^T X, \tag{30}$$

здесь

$$\vartheta_\varphi^T \Theta \vartheta'' = \vartheta_\varphi^T \chi_\vartheta^T X, \tag{31}$$

тогда

$$\vartheta_\varphi^T Fu = \vartheta_\varphi^T P, \tag{32}$$

где Fu — приведенные силы инерции по координатам ϑ,

P — приведенные внешние силы по координатам χ.

Таким образом получился рычаг Жуковского:

$$\vartheta_\varphi^T (Fu - P) = 0. \tag{33}$$

Также легко получается и метод Виттенбауэра

$$F = f(\Delta T, J). \tag{34}$$

2. Кинематика плоских механизмов

Построим механизм путем присоединения группы Ассура к кривошипу OA (рис. 1 на с. 13). Начнем с группы первого вида. Считаем, что звенья между собой соединены парой пятого класса с центром в точке B, а другими свободными элементами приводятся звеньями с заданным законом движения. В данном случае элемент кинематической пары с центром в точке A соединяется с элементом кинематической пары кривошипа. Группу Ассура представим объектом [4].

Для описания группы используем вектор объекта

$$\vartheta = (x_b, y_b, x_{sAB}, y_{sAB}, x_{sBC}, y_{sBC}, \varphi_{AB}, \varphi_{BC})^{\mathrm{T}}, \qquad (35)$$

где x_b, y_b — координаты центра кривизны элемента кинематической пары в точке B, соединяющей звенья группы;

x_{sAB}, y_{sAB}, x_{sBC}, y_{sBC} — координаты центров тяжести звеньев AB и BC;

φ_{AB}, φ_{BC} — углы поворота звеньев вокруг их центров тяжести.

2.1. Группа Ассура первого вида

На рис. 1 (с. 13) представлен объект и изображение группы Ассура первого вида.

Этот же файл можно использовать для составления длинных цепочек. Длина кривошипа OA, присоединенного к группе,

равна 0,3 м. Длины звеньев группы AB и BC равны 1,0 и 0,7 м. Расстояние от точки O вращения кривошипа до стойки в точке C равно 1,0 м.

```
## g1_bsc.py
# -*- coding: utf-8 -*-
from Tkinter import *
from bsc_grafik import *
from math import *

##============================================= Объект 1-го вида
class GA1:
  def __init__(self,AB=1.0,BC=1.0,AS=0.0,CS=0.0,alfa=0.0):
    self.AB,self.BC,self.AS,self.CS,self.alfa=AB,BC,AS,CS,alfa
    self.fi,self.xa,self.ya,self.xb,self.yb,self.xc,self.yc=0.0, 0.0,0.0, 0.0,0.0, 0.0,0.0
    self.xsAB,self.ysAB,self.xsBC,self.ysBC=0.0,0.0, 0.0,0.0
    self.fiAB,self.fiBC=0.0, 0.0
    self.bd=Bl('fi','xa','ya','xb','yb','xc','yc','xs1','ys1','xs2','ys2','fi1','fi2',
          'A','B','C','S1','S2','KL')
  def getline(self,fi):
    self.bd.add(A=(self.xa,self.ya),B=(self.xb,self.yb),C=(self.xc,self.yc),
          S1=(self.xsAB,self.ysAB),S2=(self.xsBC,self.ysBC))
    self.bd.add(fi=fi,xa=self.xa,ya=self.ya,xb=self.xb,yb=self.yb,xs1=self.
xsAB,ys1=self.ysAB,
          xs2=self.xsBC,ys2=self.ysBC,fi1=self.fiAB,fi2=self.fiBC)
  def putline(self,c,mg=0.003,xg=550,yg=200,L=['A','B','C','S1','S2'],col='black',w=1):
    for i in L: LayImagine(self.bd.L[i],mg,xg,yg,c,col,w)
  def getkontur(self,b=1):
    if b:
      self.bd.addL(KL=[(0.0,0.0),(self.xa,self.ya),(self.xb,self.yb),(self.xc,self.yc)])
    else:
      self.bd.addL(KL=[(self.xa,self.ya),(self.xb,self.yb),(self.xc,self.yc)])
    self.bd.addL(KL=F0(self.xa,self.ya,0.01))
    self.bd.addL(KL=F0(self.xb,self.yb,0.01))
    self.bd.addL(KL=F0(self.xc,self.yc,0.01))
```

```
def putkontur(self,c,mg=0.003,xg=550,yg=200,col='black',w=1):
    self.bd.showL(mg,xg,yg,c,col,w)
def position(self,xa,ya,xc,yc):
    OA2=xa**2+ya**2
    OA=OA2**0.5
    BC=self.BC
    AB=self.AB
    alfa=self.alfa
    if ya>0:   fiOA=acos((OA2+xa**2-ya**2)/(2.0*xa*OA))
    else:  fiOA=2.0*pi-acos((OA2+xa**2-ya**2)/(2.0*xa*OA))
    #-----
    AC2=(xc-xa)**2+(yc-ya)**2
    AC =AC2**0.5
    OC2=xc**2+yc**2
    OC=OC2**0.5
    if yc>0: fiOC=acos((OC2+xc**2-yc**2)/(2.0*OC*xc))
    else:   fiOC=2.0*pi-acos((OC2+xc**2-yc**2)/(2.0*OC*xc))
    #-----
    fi=fiOA-fiOC
    AK=OA*sin(fi)
    OK=OA*cos(fi)
    aco=atan(AK/(OC-OK))
    c=acos((AC2+BC**2-AB**2)/(2.0*AC*BC))
    fi3=pi-c-aco
    #-----
    xb1=OC+BC*cos(fi3)
    yb1=BC*sin(fi3)
    OB2=yb1**2+xb1**2
    OB=OB2**0.5
    boc=atan(yb1/xb1)
    fiAB=asin((yb1-AK)/AB)+alfa
    xb=OB*cos(boc+fiOC)
    yb=OB*sin(boc+fiOC)
    fiBC=-(pi-c-aco+fiOC)
    xsAB=xa+(xb-xa)/self.AB*self.AS
    ysAB=ya+(yb-ya)/self.AB*self.AS
    xsBC=xc+(xb-xc)/self.BC*self.CS
    ysBC=yc+(yb-yc)/self.BC*self.CS
    self.xa,self.ya=xa,ya
    self.xb,self.yb=xb,yb
    self.xc,self.yc=xc,yc
```

```
      self.xsAB,self.ysAB,self.xsBC,self.ysBC=xsAB,ysAB,xsBC,ysBC
      self.fiAB,self.fiBC=fiAB,fiBC
      return xb,yb,xsAB,ysAB,xsBC,ysBC,fiAB,fiBC
def show():
   OA,OC=0.4,1.0
   AB,BC=0.9,0.8
   AS,CQ=0.5,0.45
   a=pi*0.0
   xc,yc=OC*cos(a),OC*sin(a)
   g1=GA1(AB,BC,AS,CQ,a)
   mAB,mBC,JAB,JBC=1.0,1.0,1.0,1.0
   w,Eps,g=1.0,0.0,9.81
   P1,P2=mAB*g,mBC*g
   count,fcount=0,4
   du=pi/36.0
   u=0.0
   uk=2.0*pi
   bd=Bl('fi','dxb','dyb','ddxb','ddyb','JpAB','JpBC','Jp','RxB','RyB')
   while 1:
      xa,ya=OA*cos(u),OA*sin(u)
      xb,yb,xsAB,ysAB,xsBC,ysBC,fiAB,fiBC=g1.position(xa,ya,xc,yc)
      g1.getline(u)
      if count>=fcount:
         g1.getkontur()
         count=0
      count+=1
      u+=du
      if u>uk: break
   g1.bd.addL(KL=F0(xc,yc,0.01))
   g1.bd.addL(KL=F3(xc,yc,0.05,0.1))
   g1.bd.addL(KL=F0(0.0,0.0,0.01))
   g1.bd.addL(KL=F3(0.0,0.0,0.05,0.1))
   ##================================================ Labels
  bg,clr='White',['red','blue','green','black','maroon','#0DF','#C06','#A64','#684','#AC3']
   ww=[4,4,4,4,4,4,4,4,4]
   clrd=zip(clr,ww)
   Lbd =[Bd(700,55,'ПЛАН ПОЛОЖЕНИЙ',clr[3],bg)]
   Lbd+=[Bd(95, 350,"yb = [%6.3f  %6.3f]" % (min(g1.bd.L['yb']), max(g1.bd.L['yb'])),
'blue',bg)]
   Lbd+=[Bd(95, 370,"xb = [%6.3f  %6.3f]" % (min(g1.bd.L['xb']), max(g1.bd.L['xb'])),
'red',bg)]
```

```
   Lbd+=[Bd(95, 390,"fi = [%6.3f  %6.3f]" % (min(g1.bd.L['fi']), max(g1.bd.L['fi'])),
'black',bg)]
   tt ="OA=%10.3f m\nAB=%10.3f m\nBC=%10.3f m\nOC=%10.3f m\n" %
(OA,AB,BC,OC)
   tt+="CQ=%10.3f m\nalfa=%10.3f ðàä\n" % (CQ,a)
   Lbd+=[Bd(110,500,tt,'black',bg)]

   A,B=1000,650
   tk=Tk()
   tk.title('Кривошип + группа 1-го вида')
   fr=Frame(tk)
   fr.pack()
   c=Canvas(fr,bg=bg,width=A,height=B)
   c.pack(expand=1,fill=BOTH)
   mg=0.0025
   xg,yg=550,400
   for ibd in Lbd: ibd.show(c)
   g1.putline(c,mg,xg,yg,['A','B','C'],'red',3)
   g1.putline(c,mg,xg,yg,['S1','S2'],'green',5)
   g1.putkontur(c,mg,xg,yg)
   ll=[[g1.bd.L['fi'],g1.bd.L['yb']],[g1.bd.L['fi'],g1.bd.L['xb']]]
   desine_d(30,30,300,300,bg,clrd,ll,50,50,c,4,1)

   Lbd=[Bd(460,250,'A','red',bg)]
   Lbd+=[Bd(800,100,'B','red',bg)]
   Lbd+=[Bd(650,150,'S1','green',bg)]
   Lbd+=[Bd(980,220,'S2','green',bg)]
   Lbd+=[Bd(920,400,'C','black',bg)]
   Lbd+=[Bd(550,460,'O','black',bg)]
   for ibd in Lbd: ibd.show(c)

   tk.mainloop()
if __name__=="__main__": show()
```

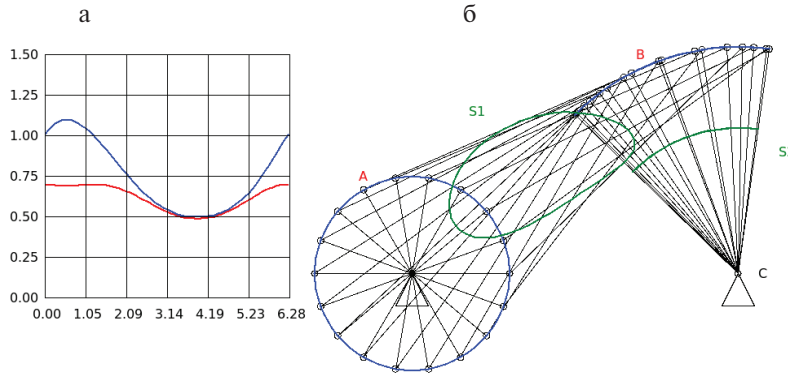


Рис. 1. Механизм с группой Ассура первого вида:
а — координаты *x* и *y* точки B; б — план положений

2.2. Группа Ассура второго вида

Группа Ассура второго вида отличается тем, что на конце второго звена находится элемент кинематической пары поступательного вида. Кривошип задает движение точке A, а ползун, наклон и положение которого задается третьим звеном, с точкой C на ползуне устанавливается на направляющей (рис. 2 на с. 17).

Для определения положения звеньев группы используем вектор

$$\vartheta = (x_b, y_b, x_{sAB}, y_{sAB,} \varphi_{AB}), \qquad (36)$$

где x_b, y_b — координаты центра кривизны элемента кинематической пары в точке B, соединяющей звенья группы;

x_{sAB}, y_{sAB} — координаты центра тяжести звена AB;

φ_{AB} — угол поворота звена AB вокруг центра тяжести.

Ниже приведена программа с объектом второй группы. Длина кривошипа OA = 0,25 м, шатуна AB = 0,7 м, угол наклона направляющей 0,262°, дезаксиал равен нулю.

```
## g2_bsc.py
# -*- coding: utf-8 -*-
from Tkinter import *
from bsc_grafik import *
from math import *

##========================================== Объект 2-го вида
class GA2:
  def __init__(self,AB=1.0,e=0.0,alfa=0.0,AS=0.0):
    self.AB,self.e,self.alfa,self.AS=AB,e,alfa,AS
    self.xsAB,self.ysAB,self.fiAB=0.0,0.0,0.0
    self.xa,self.ya,self.xb,self.yb=0.0,0.0, 0.0,0.0
    self.bd=Bl('fi','xa','ya','xs1','ys1','xb','yb','fi1','A','B','S1','KL')
  def getline(self,fi):
    self.bd.add(A=(self.xa,self.ya),B=(self.xb,self.yb),S1=(self.xsAB,self.ysAB))
    self.bd.add(fi=fi,xa=self.xa,ya=self.ya,xb=self.xb,yb=self.yb,xs1=self.
xsAB,ys1=self.ysAB,
                fi1=self.fiAB)
  def putline(self,c,mg=0.003,xg=550,yg=200,L=['A','B','S1'],col='black',w=1):
    for i in L: LayImagine(self.bd.L[i],mg,xg,yg,c,col,w)
  def getkontur(self,b=1):
    if b:
      self.bd.addL(KL=[(0.0,0.0),(self.xa,self.ya),(self.xb,self.yb)])
    else:
      self.bd.addL(KL=[(self.xa,self.ya),(self.xb,self.yb)])
    self.bd.addL(KL=F0(self.xa,self.ya,0.01))
    self.bd.addL(KL=F0(self.xb,self.yb,0.01))
    self.bd.addL(KL=F4(self.xb,self.yb,0.03,0.02,self.alfa))
  def putkontur(self,c,mg=0.003,xg=550,yg=200,col='black',w=1):
    self.bd.showL(mg,xg,yg,c)
  def position(self,xa,ya):
    e=self.e
    alfa=self.alfa
    OA2=xa**2+ya**2
    OA=OA2**0.5
    if ya>0:    fiOA=acos((OA2+xa**2-ya**2)/(2.0*xa*OA))
    else: fiOA=2.0*pi-acos((OA2+xa**2-ya**2)/(2.0*xa*OA))
    #-----
```

```
        fi=fiOA-alfa
        OK=OA*cos(fi)
        AK=OA*sin(fi)
        xb1=OK+(self.AB**2-(AK-e)**2)**0.5
        yb1=e
        OB2=yb1**2+xb1**2
        OB=OB2**0.5
        if yb1>0:   fiOB=acos((OB2+xb1**2-yb1**2)/(2.0*xb1*OB))
        else: fiOB=2.0*pi-acos((OB2+xb1**2-yb1**2)/(2.0*xb1*OB))
        xb=OB*cos(alfa+fiOB)
        yb=OB*sin(alfa+fiOB)
        #-----
        fiAB=asin((yb1-AK)/self.AB)+alfa
        xsAB=xa+(xb-xa)/self.AB*self.AS
        ysAB=ya+(yb-ya)/self.AB*self.AS

        self.xa,self.ya,self.xb,self.yb=xa,ya,xb,yb
        self.xsAB,self.ysAB=xsAB,ysAB
        self.fiAB=fiAB
        return xb,yb,xsAB,ysAB,fiAB
##================================================ Calculating
def show():
    OA,OC=0.3,1.0
    AB=1.0
    AS=0.4
    a=pi/6.0
    e=0.2
    g2=GA2(AB,e,a,AS)
    count,fcount=0,4
    du=pi/36.0
    u=0.0
    uk=2.0*pi
    bd=Bl('fi','dxb','dyb','ddxb','ddyb','JpAB','JpB','Jp','RxA','RyA','RxB','RyB','N')
    while 1:
        xa,ya=OA*cos(u),OA*sin(u)
        xb,yb,xsAB,ysAB,fiAB=g2.position(xa,ya)
        g2.getline(u)
        if count>=fcount:
            g2.getkontur()
            count=0
        count+=1
```

```
      u+=du
      if u>uk: break
   g2.bd.addL(KL=F0(0.0,0.0,0.01))
   g2.bd.addL(KL=F3(0.0,0.0,0.05,0.1))
   ##================================================ Labels
   crl=["black","red","green","#0DF","#C06","#A64","#686","blue","#AC3"]
   bg='White'
   Lbd =[Bd(600,55,'ПЛАН ПОЛОЖЕНИЙ',crl[6],bg)]
   Lbd+=[Bd(95, 350,"yb = [%6.3f  %6.3f]" % (min(g2.bd.L['yb']), max(g2.
bd.L['yb'])),'blue',bg)]
   Lbd+=[Bd(95, 370,"xb = [%6.3f  %6.3f]" % (min(g2.bd.L['xb']), max(g2.
bd.L['xb'])),'red',bg)]
   Lbd+=[Bd(95, 390,"fi = [%6.3f  %6.3f]" % (min(g2.bd.L['fi']), max(g2.
bd.L['fi'])),'black',bg)]
   tt ="OA=%10.3f m\nAB=%10.3f m\n" % (OA,AB)
   tt+="e=%10.3f m\nAS=%10.3f m\n" % (e,AS)
   tt+="alfa=%10.3f m\n" % (a)
   Lbd+=[Bd(120,500,tt,"black",bg)]

   ##================================================ Tkinter

   A,B=1000,680
   tk=Tk()
   tk.title('Кривошип + группа 2-го вида')
   fr=Frame(tk)
   fr.pack()
   c=Canvas(fr,bg=bg,width=A,height=B)
   c.pack(expand=1,fill=BOTH)
   bg,clr='White',['red','blue',"black",'green','red','blue',"black",'green','red','blue',
            "black",'green','maroon',"#0DF","#C06","#A64","#684","#AC3"]
   ww=[4,2,2,1,1,1,1,1,1,1,1,1,1,1,1,1]
   clrd=zip(clr,ww)
   ll=[[g2.bd.L['fi'],g2.bd.L['yb']],[g2.bd.L['fi'],g2.bd.L['xb']]]
   desine_d(30,30,300,300,bg,clrd,ll,50,50,c,4,1)
   mg,xg,yg=0.002,450,500
   for ibd in Lbd: ibd.show(c)
   g2.putline(c,mg,xg,yg,['S1'],'green',3)
   g2.putline(c,mg,xg,yg,['A','B'],'blue',3)
   g2.putkontur(c,mg,xg,yg)
   #=================================================
```

```
    Lbd=[Bd(410,330,'A','blue',bg)]
    Lbd+=[Bd(900,90,'B','blue',bg)]
    Lbd+=[Bd(600,250,'S1','green',bg)]
    Lbd+=[Bd(450,570,'O','black',bg)]
    for ibd in Lbd: ibd.show(c)
    tk.mainloop()
if __name__=="__main__": show()
```

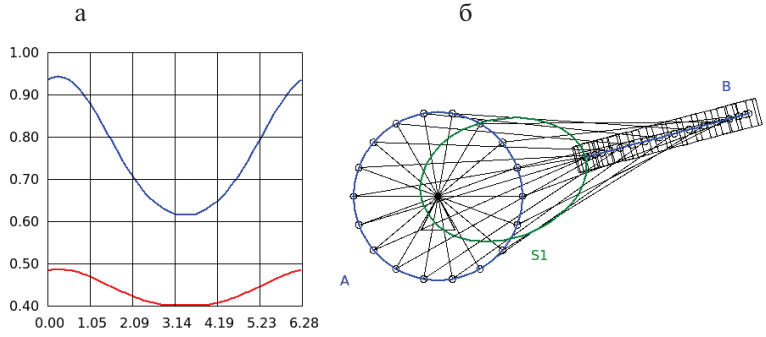


Рис. 2. Механизм с группой Ассура второго вида:
а — координаты *x* и *y* точки B; б — план положений

2.3. Группа Ассура третьего вида

В группе третьего вида звенья соединены поступательной парой, ее направляющая устанавливается по ползуну, являющемся первым звеном группы, соединенным с ведущим кривошипом (рис. 3 на с. 22).

Группа описана вектором

$$\vartheta = (x_b, y_b, x_d, y_d, x_{sBD}, y_{sBD}, \varphi_{BD}), \qquad (37)$$

в котором x_b, y_b — координаты центра кривизны элемента кинематической пары в точке B, соединяющей звенья группы;

x_d, y_d — координаты центра кривизны элемента кинематической пары в точке D;

x_{sBD}, y_{sBD} — координаты центра тяжести звена BD;

φ_{BD} — угол поворота звена BD вокруг центра тяжести.

Ниже приведен механизм с группой Ассура третьего вида. Длина кривошипа OA = 0,3 м, длина кулисы BD = 1,3 м, консоль BC = 0,25 м. Отрезок OC = 0,2 м наклонен под углом 0,942°.

```
## g3_bsc.py
# -*- coding: utf-8 -*-
from Tkinter import *
from bsc_grafik import *
from math import *
##================================================ Parameter
class GA31:
  def __init__(self,BD=1.0,BC=0.0,BS=0.0):
    self.BD,self.BC,self.BS=BD,BC,BS
    self.xa,self.ya,self.alfa=0.0,0.0,0.0
    self.xb,self.yb,self.xd,self.yd,self.xsBD,self.ysBD,self.fiBD=0.0,0.0,0.0,0.0,0.0,0.0,0
.0
    self.bd=Bl('fi','xa','ya','xb','yb','xd','yd','xs1','ys1','fi1','A','B','D','S1','KL')
  def getline(self,fi):
    self.bd.add(A=(self.xa,self.ya),B=(self.xb,self.yb),D=(self.xd,self.yd),
          S1=(self.xsBD,self.ysBD))
    self.bd.add(fi=fi,xa=self.xa,ya=self.ya,xb=self.xb,yb=self.yb,xs1=self.xsBD,
          ys1=self.ysBD,fi1=self.fiBD,xd=self.xd,yd=self.yd)
  def putline(self,c,mg=0.003,xg=550,yg=200,L=['A','B','S1'],col='black',w=1):
    for i in L: LayImagine(self.bd.L[i],mg,xg,yg,c,col,w)
  def getkontur(self,b=1):
    if b:
      self.bd.addL(KL=[(0.0,0.0),(self.xa,self.ya),(self.xb,self.yb),(self.xc,self.yc)])
```

```
      else:
         self.bd.addL(KL=[(self.xa,self.ya),(self.xb,self.yb),(self.xc,self.yc)])
      self.bd.addL(KL=[(self.xa,self.ya),(self.xd,self.yd)])
      self.bd.addL(KL=F0(self.xa,self.ya,0.01))
      self.bd.addL(KL=F4(self.xa,self.ya,0.05,0.02,self.fiBD))
   def putkontur(self,c,mg=0.003,xg=550,yg=200,col='black',w=1):
      self.bd.showL(mg,xg,yg,c,col,w)
   def position(self,xa,ya,xc,yc):
      self.xa,self.ya=xa,ya
      self.xc,self.yc=xc,yc
      OA2=xa**2+ya**2
      OA=OA2**0.5
      if ya>0: fiOA=acos((OA2+xa**2-ya**2)/(2.0*xa*OA))
      else:   fiOA=2.0*pi-acos((OA2+xa**2-ya**2)/(2.0*xa*OA))
      OC2=xc**2+yc**2
      OC=OC2**0.5
      if yc>0: fiOC=acos((OC2+xc**2-yc**2)/(2.0*OC*xc))
      else:   fiOC=2.0*pi-acos((OC2+xc**2-yc**2)/(2.0*OC*xc))
      AC2=(xa-xc)**2+(ya-yc)**2
      AC=AC2**0.5
      AB2=AC2-self.BC**2
      AB=AB2**0.5
      #-----
      dx,dy=xa-xc,ya-yc
      if dy>0: gamma=acos((AC2+dx**2-dy**2)/(2.0*AC*dx))
      else:   gamma=2.0*pi-acos((AC2+dx**2-dy**2)/(2.0*AC*dx))
      alfa=atan(self.BC/AB)
      fiBD=alfa+gamma
      xb=xa+AB*cos(fiBD+pi)
      yb=ya+AB*sin(fiBD+pi)
      xd=xa+(self.BD-AB)*cos(fiBD)
      yd=ya+(self.BD-AB)*sin(fiBD)
      #-----
      xsBD=xb+self.BS*cos(fiBD)
      ysBD=yb+self.BS*sin(fiBD)

      self.xb,self.yb,self.xd,self.yd,self.xsBD,self.ysBD,self.fiBD,
      self.alfa=xb,yb,xd,yd,xsBD,ysBD,fiBD,alfa
      return xb,yb,xd,yd,xsBD,ysBD,fiBD
def show():
   OA,OC=0.45,0.95
```

```
alfa=-pi*0.1
xc1=OC*cos(alfa)
yc1=OC*sin(alfa)
#---
OC3=-0.2
BD31=1.5
BC31=-0.3
BS31=1.0
g31=GA31(BD31,BC31,BS31)
##================================================ Calculating
du,u,uk=pi/36.0,0.0,2.0*pi
count,fcount=0,4
while 1:
   xa,ya=OA*cos(u),OA*sin(u)
   g31.position(xa,ya,xc1,yc1)
   g31.getline(u)
   if count>=fcount:
      g31.getkontur()
      count=0
   count+=1
   u+=du
   if u>uk: break
g31.bd.addL(KL=F0(xc1,yc1,0.01))
g31.bd.addL(KL=F3(xc1,yc1,0.05,0.1))
g31.bd.addL(KL=F0(0.0,0.0,0.01))
g31.bd.addL(KL=F3(0.0,0.0,0.05,0.1))
##================================================== Labels
crl=["blue","red","green","#0DF","#C06","#A64","#686","blue","#AC3"]
bg='White'
Lbd =[Bd(500,25,'ПЛАН ПОЛОЖЕНИЙ',crl[6],bg)]
tt ="xd= [%6.3f   %6.3f]\n" % (min(g31.bd.L['xd']),max(g31.bd.L['xd']))
Lbd+=[Bd(100,360,tt,'blue',bg)]
tt ="yd= [%6.3f   %6.3f]\n" % (min(g31.bd.L['yd']),max(g31.bd.L['yd']))
Lbd+=[Bd(100,380,tt,'red',bg)]
tt ="fi= [%6.3f   %6.3f]\n" % (min(g31.bd.L['fi']),max(g31.bd.L['fi']))
Lbd+=[Bd(100,400,tt,'black',bg)]
tt ="OA=%10.3f m\nOC=%10.3f m\n" % (OA,OC)
tt+="OC3=%10.3f m\nBD31=%10.3f m\n" % (OC3,BD31)
tt+="BC31=%10.3f m\nBS31=%10.3f m\n" % (BC31,BS31)
tt+="alfa=%10.3f рад\n" % (alfa)
Lbd+=[Bd(110,500,tt,'black',bg)]
```

```
    ##================================================ Tkinter
    A,B=1000,700
    tk=Tk()
    tk.title('Кривошип + группа 1-,2-,3-го вида')
    fr=Frame(tk)
    fr.pack()
    c=Canvas(fr,bg=bg,width=A,height=B)
    c.pack(expand=1,fill=BOTH)
    #================================================ Output
    bg,clr='White',['red','blue',"black",'green','red','blue',"black",'green','red','blue',
               "black",'green','maroon',"#0DF","#C06","#A64","#684","#AC3"]
    ww=[4,2,2,1,1,1,1,1,1,1,1,1,1,1,1,1]
    ll=[[g31.bd.L['fi'],g31.bd.L['yd']],[g31.bd.L['fi'],g31.bd.L['xd']]]
    clrd=zip(clr,ww)
    desine_d(30,30,300,300,bg,clrd,ll,50,50,c,4,1)
    for ibd in Lbd: ibd.show(c)
    mg,xg,yg=0.003,600,400
    g31.putline(c,mg,xg,yg,['A','B','D'],'blue',3)
    g31.putline(c,mg,xg,yg,['S1'],'green',3)
    g31.putkontur(c,mg,xg,yg)

    Lbd=[Bd(740,300,'A','blue',bg)]
    Lbd+=[Bd(850,610,'B','blue',bg)]
    Lbd+=[Bd(710,200,'S3','green',bg)]
    Lbd+=[Bd(910,490,'C','black',bg)]
    Lbd+=[Bd(410,250,'D','blue',bg)]
    Lbd+=[Bd(590,450,'O','black',bg)]
    for ibd in Lbd: ibd.show(c)

    tk.mainloop()
if __name__=="__main__": show()
```

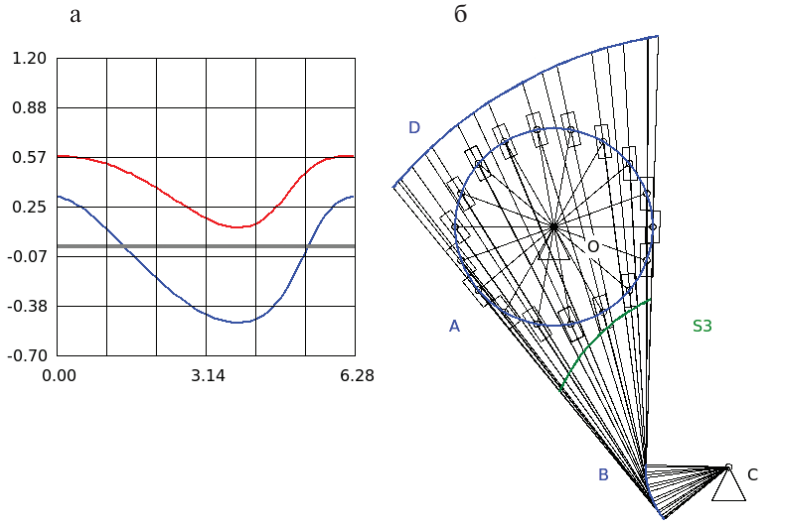


Рис. 3. Механизм с группой Ассура третьего вида:
а — координаты *x* и *y* точки D; б — план положений

Возможен другой вариант группы третьего вида. Соединим с ведущим кривошипом одну из точек направляющей, а ползун второго звена будет вращаться относительно точки B третьего звена (рис. 4):

$$\vartheta = (x_a, y_a, x_b, y_b, x_c, y_c, x_{sAC}, y_{sAC}, \varphi_{AC}), \qquad (38)$$

здесь x_a, y_a — координаты центра кривизны элемента кинематической пары в точке A;

x_b, y_b — координаты центра кривизны элемента кинематической пары в точке B;

x_c, y_c — координаты центра кривизны элемента кинематической пары в точке C;

x_{sAC}, y_{sAC} — координаты центра тяжести звена AC;

φ_{AC} — угол поворота звена AC вокруг центра тяжести.

Длина кривошипа OA = 0,3 м, длина шатуна AC = 1,0 м, длина неподвижного отрезка OB = 0,5 м.

```
## g4_bsc.py
# -*- coding: utf-8 -*-
from Tkinter import *
from bsc_grafik import *
from math import*
##================================================ Parameter
class GA32:
   def __init__(self,AC=1.0,AS=0.0,alfa=0.0):
      self.AC,self.AS=AC,AS
      self.xa,self.ya=0.0,0.0
      self.xb,self.yb=0.0,0.0
      self.xc,self.yc=0.0,0.0
      self.fi,self.AB=0.0,0.0
      self.alfa=alfa
      self.xsAC,self.ysAC,self.fiAC=0.0,0.0,0.0
      self.bd=Bl('fi','xa','ya','xb','yb','xc','yc','xs1','ys1','fi1','AB','S1','A','B','C','KL')
   def getline(self,fi):
      self.fi=fi
      self.bd.add(A=(self.xa,self.ya),B=(self.xb,self.yb),C=(self.xc,self.yc),
                  S1=(self.xsAC,self.ysAC))
      self.bd.add(fi=fi,xa=self.xa,ya=self.ya,xb=self.xb,yb=self.yb,
                   xs1=self.xsAC,ys1=self.ysAC,fi1=self.fiAC,AB=self.AB)
   def putline(self,c,mg=0.003,xg=550,yg=200,L=['A','C','S1'],col='black',w=1):
      for i in L: LayImagine(self.bd.L[i],mg,xg,yg,c,col,w)
   def getkontur(self):
      self.bd.addL(KL=[(0.0,0.0),(self.xa,self.ya),(self.xc,self.yc)])
      self.bd.addL(KL=F0(self.xa,self.ya,0.01))
      self.bd.addL(KL=F4(self.xb,self.yb,0.05,0.02,self.fiAC))
   def putkontur(self,c,mg=0.003,xg=550,yg=200,col='black',w=1):
      self.bd.showL(mg,xg,yg,c,col,w)
   def position(self,xa,ya,xb,yb):
      self.xa,self.ya=xa,ya
      self.xb,self.yb=xb,yb
      AB=pow((ya-yb)**2+(xa-xb)**2,0.5)
      self.AB=AB
      dx,dy=xb-xa,yb-ya
      a=abs(dy/dx)
      if dy>0 and dx>0:   # 1
```

```
        if a<1:       gamma=atan(a)
        else:         gamma=0.5*pi-atan(1.0/a)
    elif dy>0 and dx==0: gamma=0.5*pi
    elif dy>0 and dx<0: # 2
        if a<1:       gamma=pi-atan(a)
        else:         gamma=0.5*pi+atan(1.0/a)
    elif dy<0 and dx>0: # 4
        if a<1:       gamma=-atan(a)
        else:         gamma=-0.5*pi+atan(1.0/a)
    elif dy<0 and dx==0: gamma=-0.5*pi
    elif dy<0 and dx<0: # 3
        if a<1:       gamma=-pi+atan(a)
        else:         gamma=-0.5*pi-atan(1.0/a)
    elif dy==0 and dx<0: gamma=-pi
    elif dy==0 and dx>0: gamma=0.0
    else: gamma=atan(a)

    xc=xa+(xb-xa)/AB*self.AC
    yc=ya+(yb-ya)/AB*self.AC
    xs=xa+(xb-xa)/AB*self.AS
    ys=ya+(yb-ya)/AB*self.AS
    self.xc,self.yc,self.xsAC,self.ysAC,self.fiAC=xc,yc,xs,ys,gamma
    return xa,ya,xb,yb,xc,yc,self.xsAC,self.ysAC,self.fiAC

def show():
  OA,OB,AC,AS=0.3,0.5,1.0,0.35
  alfa=pi*0.15
  g32=GA32(AC,AS)
  du,u,uk=pi/36.0,0.0,2.0*pi
  xb,yb=OB*cos(alfa),OB*sin(alfa)
  ##=============================================== Calculating
  du,u,uk=pi/36.0,0.0,2.0*pi
  count,fcount=0,3
  while 1:
    fi=u
    ya,xa=OA*sin(fi),OA*cos(fi)
    g32.position(xa,ya,xb,yb)
    g32.getline(u)
    if count>=fcount:
      g32.getkontur()
      count=0
```

```
      count+=1
      u+=du
      if u>uk: break
   g32.bd.addL(KL=F0(0.0,0.0,0.01))
   g32.bd.addL(KL=F3(0.0,0.0,0.05,0.1))
   g32.bd.addL(KL=F0(xb,yb,0.01))
   g32.bd.addL(KL=F3(xb,yb,0.05,0.1))
   ##================================================ Labels
   crl=["blue","red","green","#0DF","#C06","#A64","#686","blue","#AC3"]
   bg='White'
   Lbd =[Bd(500,25,'ПЛАН ПОЛОЖЕНИЙ',crl[6],bg)]
   tt ="ys = [%6.3f   %6.3f]\n" % (min(g32.bd.L['ys1']), max(g32.bd.L['ys1']))
   Lbd+=[Bd(130,360,tt,'blue',bg)]
   tt ="xs = [%6.3f   %6.3f]\n" % (min(g32.bd.L['xs1']), max(g32.bd.L['xs1']))
   Lbd+=[Bd(130,380,tt,'red',bg)]
   tt ="fi = [%6.3f   %6.3f]\n" % (min(g32.bd.L['fi']), max(g32.bd.L['fi']))
   Lbd+=[Bd(130,400,tt,'black',bg)]
   tt ="OA=%10.3f m\nOB=%10.3f m\nAC=%10.3f m\nAS=%10.3f m\n" %
(OA,OB,AC,AS)
   Lbd+=[Bd(110,500,tt,'black',bg)]
   ##================================================ Tkinter
   A,B=1000,700
   tk=Tk()
   tk.title('Кривошип + группа 3-го вида')
   fr=Frame(tk)
   fr.pack()
   c=Canvas(fr,bg=bg,width=A,height=B)
   c.pack(expand=1,fill=BOTH)
   #================================================ Output
   bg,clr='White',['red','blue','black','green','red','blue','black','green','red','blue',
            'black','green','maroon',"#0DF","#C06","#A64","#684","#AC3"]
   ww=[4,2,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1]
   ll=[[g32.bd.L['fi'],g32.bd.L['xs1']],[g32.bd.L['fi'],g32.bd.L['ys1']]]
   clrd=zip(clr,ww)
   desine_d(30,30,300,300,bg,clrd,ll,50,50,c,4,1)
   for ibd in Lbd: ibd.show(c)
   mg,xg,yg=0.0025,500,450
   g32.putline(c,mg,xg,yg,['A','C'],'blue',3)
   g32.putline(c,mg,xg,yg,['S1'],'green',3)
   g32.putkontur(c,mg,xg,yg)
```

```
    Lbd=[Bd(500,300,'A','blue',bg)]
    Lbd+=[Bd(680,415,'B','black',bg)]
    Lbd+=[Bd(620,330,'S','green',bg)]
    Lbd+=[Bd(910,420,'C','blue',bg)]
    Lbd+=[Bd(500,510,'O','black',bg)]
    for ibd in Lbd: ibd.show(c)

    tk.mainloop()
if __name__=="__main__": show()
```

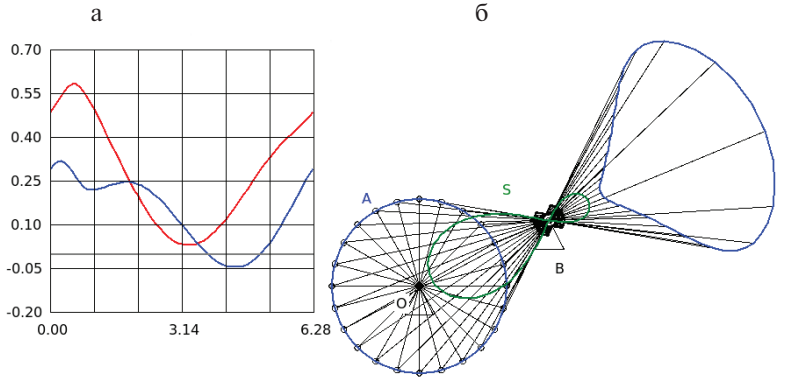


Рис. 4. Механизм с группой Ассура третьего вида:
а — координаты *x* и *y* точки *S* по углу поворота кривошипа;
б — план положений

3. Кулачковые механизмы

Для построения профиля кулака используется обращенное движение (рис. 5 на с. 29). Вместо кривошипа вращается кулачковая шайба, приводящая в движение толкатель. Возвратно-поступательное движение толкателя зависит от размеров кулака и других параметров кулачковой передачи

$$\vartheta = (x_a, y_a, x_b, y_b, x_c, y_c, x_{sAC}, y_{sAC}, \varphi_{AC}). \quad (39)$$

Радиус ролика 0,25 м, дезаксиал 0,04 м, минимальный радиус кулачковой шайбы 0,35 м.

```
## klp_bsc.py
# -*- coding: utf-8 -*-
from Tkinter import *
from bsc_grafik import *
from math import *
##================================================ Parameter
r,fi,rr=0.25,0.0,0.04
dfi=pi/24.0
fip=2.0*pi-dfi
h_max=0.05
xo,yo=0.0,0.0
R=r+h_max*2.0
e=0.1
```

```
##================================================ Calculating
bl=Bl('fi','xa','ya','xo1','yo1','h','A','KL')
while fi<fip:
   h=h_max*sin(fi)
   ro=r+h
   xe=xo+e*cos(fi+pi/2.0)
   ye=yo+e*sin(fi+pi/2.0)
   xa=xe+ro*cos(fi)
   ya=ye+ro*sin(fi)
   x=xe+R*cos(fi)
   y=ye+R*sin(fi)
   #----------
   bl.add(fi=fi,xa=xa,ya=ya,h=h)
   bl.add(A=(xa,ya))
   bl.addL(KL=F0(xa,ya,rr))
   bl.addL(KL=F0(xa,ya,0.005))
   bl.addL(KL=[(xo,yo),(xe,ye),(x,y)])
   bl.addL(KL=F4(xa,ya,0.02,0.01,fi))
   fi+=dfi
bl.addL(KL=F0(0.0,0.0,0.007))
bl.addL(KL=F3(0.0,0.0,0.015,0.04))
##=================================================== Labels
crl=["black","red","green","#0DF","#C06","#A64","#686","blue","#AC3"]
ww=[2,2,2,2,2,2,2,2]
crld=zip(crl,ww)
bg='White'
Lbd =[Bd(800,55,'ПЛАН ПОЛОЖЕНИЙ',crl[6],bg)]
Lbd+=[Bd(85, 300,"ya = [%6.3f  %6.3f]" % (min(bl.L['ya']), max(bl.L['ya'])), crl[0],bg)]
Lbd+=[Bd(85, 325,"xa = [%6.3f  %6.3f]" % (min(bl.L['xa']), max(bl.L['xa'])), crl[1],bg)]
tt ="ya = [%6.3f  %6.3f]\n" % (min(bl.L['ya']), max(bl.L['ya']))
tt +="xa = [%6.3f  %6.3f]" % (min(bl.L['xa']), max(bl.L['xa']))
Lbd+=[Bd(85, 600,tt,crl[0],bg)]
tt ="fin=%10.3f r\nfik=%10.3f r\n" % (min(bl.L['fi']),max(bl.L['fi']))
tt+="R=%10.3f m\nr=%10.3f m\nBC=%10.3f m\n" % (R,r,rr)
Lbd+=[Bd(410,640,tt,crl[4],bg)]
##=================================================== Tkinter
A,B=1000,700
A,B=1000,700
tk=Tk()
tk.title('Кулак с линейно движущимся толкателем')
fr=Frame(tk)
```

```
fr.pack()
c=Canvas(fr,bg='White',width=A,height=B)
c.pack(expand=1,fill=BOTH)
#=================================================== Output
mg,xg,yg=0.0012,680,380
desine_l(20,30,300,300,(255,255,255),crld,bl.L['fi'],[bl.L['xa'],bl.L['ya']],0,50,50,c)
desine_l(20,360,300,300,(255,255,255),crld,bl.L['xa'],[bl.L['ya']],0,50,50,c)
for ibd in Lbd: ibd.show(c)
LayImagine(bl.L['A'],mg,xg,yg,c)
bl.showLk('KL',mg,xg,yg,c)
#put_shelve('Lbd',Lbd,'bdkl.dat')
tk.mainloop()
```

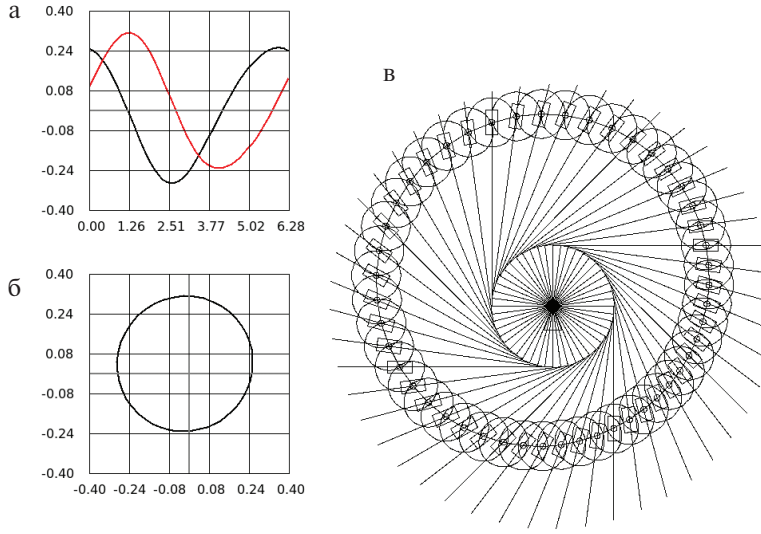


Рис. 5. Кулачковый механизм с поступательным движением толкателя:
а — координаты *x* и *y* текущей точки контакта по углу поворота кривошипа;
б — координата *y* текущей точки контакта по координате *x*; в — план положений

Кулачковый механизм (рис. 6 на с. 33) отличается только соединением толкателя со стойкой вращательной парой

$$\vartheta = (x_a, y_a, x_b, y_b, x_c, y_c, x_{sAC}, y_{sAC}, \varphi_{AC})^{\mathrm{T}}, \qquad (39)$$

такой механизм работает при замкнутой кинематической паре четвертого класса.

Радиус ролика 0,25 м, расстояние между неподвижными точками 0,4 м.

```
## krp_bsc.py
# -*- coding: utf-8 -*-
from Tkinter import *
from bsc_grafik import *
from math import*

##================================================ Parameter
a,r,fi,rr=0.25,0.25,0.0,0.04
dfi=pi/24.0
fip=2.0*pi+dfi
alfa=pi/4.0
fi_max=pi/10.0
##================================================ Calculating
bl=Bl('fi','xa','ya','xo1','yo1','fikp','A','O1','KL')
br=Bl('A','KR')
bk=Bl('A','KK')
xo=0.0
yo=0.0
while fi<fip:
   fikp=fi_max*sin(fi)
   ro=sqrt(a*a+r*r-2.0*a*r*cos(alfa+fikp))
   b=acos((ro*ro+a*a-r*r)/(2.0*ro*a))
   xa=xo+ro*cos(fi+b)
   ya=yo+ro*sin(fi+b)
   #----------
```

```
    xas=xo+ro*cos(b)
    yas=yo+ro*sin(b)
    #----------
    xo1=a*cos(fi)
    yo1=a*sin(fi)
    bl.add(fi=fi,xa=xa,ya=ya,fikp=fikp,A=(xa,ya),O1=(xo1,yo1))
    bl.addL(KL=[(xa,ya),(xo1,yo1),(xo,yo)])
    bl.addL(KL=F0(xa,ya,0.005))
    bl.addL(KL=F0(xa,ya,rr))
    bl.addL(KL=F0(xo1,yo1,0.005))
    #----------
    br.add(A=(xas,yas))
    br.addL(KR=[(xas,yas),(a,0.0)])
    br.addL(KR=F0(xas,yas,0.005))
    #----------
    bk.add(A=(xa,ya))
    bk.addL(KK=F0(xa,ya,rr))
    bk.addL(KK=F0(xa,ya,0.005))
    bk.addL(KK=[(xa,ya),(xo,yo)])
    fi+=dfi
bl.addL(KL=F3(xo,yo,0.015,0.04))
bl.addL(KL=F0(0.0,0.0,0.007))
br.addL(KR=F3(a,0.0,0.015,0.04))
br.addL(KR=F0(a,0.0,0.005))
bk.addL(KK=F0(0.0,0.0,0.007))
bk.addL(KK=F3(0.0,0.0,0.015,0.04))
##=================================================== Labels
crl=["red","green","#0DF","#C06","#A64","#686","blue","#AC3"]
ww=[2,2,2,2,2,2,2,2]
crld=zip(crl,ww)
bg='White'
Lbd =[Bd(800,55,'ПЛАН ПОЛОЖЕНИЙ',crl[6],bg)]
Lbd+=[Bd(260, 295,"ya = [%6.3f  %6.3f]" % (min(bl.L['ya']), max(bl.L['ya'])),
crl[0],bg)]
Lbd+=[Bd(260, 315,"xa = [%6.3f  %6.3f]" % (min(bl.L['xa']), max(bl.L['xa'])), crl[1],bg)]

tt ="ya = [%6.3f  %6.3f]\n" % (min(bl.L['ya']), max(bl.L['ya']))
tt+="xa = [%6.3f  %6.3f]" % (min(bl.L['xa']), max(bl.L['xa']))
Lbd+=[Bd(85, 615,tt,'black',bg)]

tt ="fin=%10.3f r\nfik=%10.3f r\n" % (min(bl.L['fi']),max(bl.L['fi']))
```

```
tt+="a=%10.3f m\nr=%10.3f m\nBC=%10.3f m\n" % (a,r,rr)
Lbd+=[Bd(410,600,tt,'black',bg)]
tt ="xa = [%6.3f   %6.3f]\n" % (min(bl.L['xa']), max(bl.L['xa']))
tt+="ya = [%6.3f   %6.3f]\n" % (min(bl.L['ya']), max(bl.L['ya']))
Lbd+=[Bd(850,100,tt,crl[-1],bg)]
##================================================ Tkinter
A,B=1000,700
tk=Tk()
tk.title('Кулак с поворотным толкателем')
fr=Frame(tk)
fr.pack()
c=Canvas(fr,bg='White',width=A,height=B)
c.pack(expand=1,fill=BOTH)
#================================================= Output
mg,xg,yg=0.0035,450,250
crl=["red","green","#ODF","#CO6","#A64","#686","AC3"]
mg,xg,yg=0.0015,550,200

desine_l(20,30,300,300,(255,255,255),crld,bl.L['fi'],[bl.L['xa'],bl.L['ya']],0,50,50,c)
desine_l(20,360,300,300,(255,255,255),[('blue',3)],bl.L['xa'],[bl.L['ya']],0,50,50,c)

bl.showL(mg,xg,yg,c)
LayImagine(bl.L['A'],mg,xg,yg,c)
LayImagine(bl.L['O1'],mg,xg,yg,c)
#-------------
LayImagine(bk.L['A'],mg,xg,yg,c)
bk.showLk('KK',mg,xg+300,yg+350,c)
#-------------
br.showLk('KR',mg,xg-150,yg+380,c)
LayImagine(br.L['A'],mg,xg-150,yg+380,c)
#-------------
for ibd in Lbd: ibd.show(c)
#put_shelve('Lbd',Lbd,'bdkr.dat')
tk.mainloop()
```

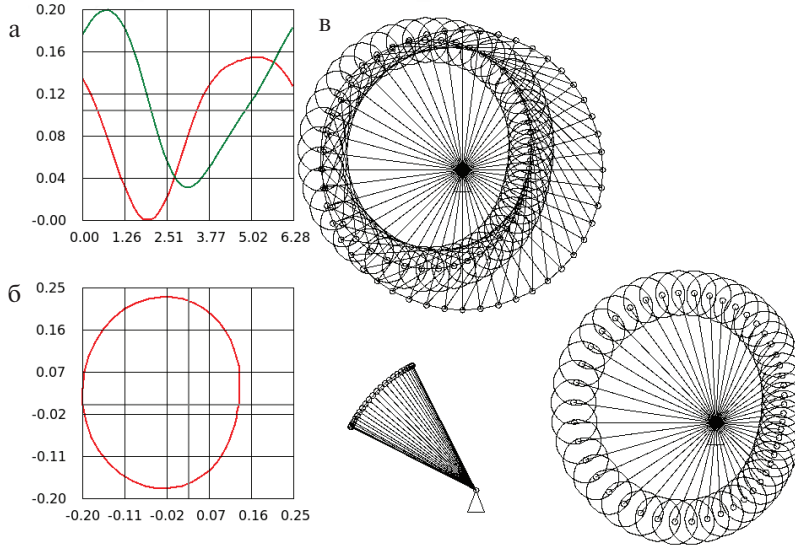


Рис. 6. Кулачковый механизм с поворотным движенем толкателя:

а — координаты *x* и *y* текущей точки контакта по углу поворота кривошипа; б — координата *y* текущей точки контакта по координате *x*; в — план положений

4. Зубчато-рычажные механизмы

Рассмотрим только один пример механизма такого типа. Пусть на конце кривошипа вращается сателлит, находящийся в зацеплении с неподвижным эпициклом. При этом любая точка на сателлите совершает некоторую замкнутую траекторию (рис. 7 на с. 36), если отношение чисел зубьев эпицикла к сателлиту кратно.

Отличие механизма на рис. 7 в том, что на сателлите имеется вынос длиной 0,25 м и передаточное число между сателлитом и эпициклом равно двум.

```
## mpp_bsc.py
# -*- coding: utf-8 -*-
from Tkinter import *
from bsc_grafik import *
from math import *
bd=Bl('A','B','Ax','Ay','Bx','By','fi1','fi2','alf','KL')
bk=Bl('KL')
##================================================ Parameter
m,i=0.05,3
d=0.05
z3=60.0
z2=z3/i
a=0.5*(z3-z2)*m
```

```
h=z2*m*0.5+d
dfi1,fi1=pi/20.0,0.0
fi2=(1-z3/z2)*fi1
while 1:
   Ax,Ay=a*cos(fi1),a*sin(fi1)
   Bx,By=Ax+h*cos(fi2),Ay+h*sin(fi2)
   AB=sqrt(Bx**2+By**2)
   alf=abs(fi1-fi2)
   bd.add(fi1=fi1,fi2=fi2,Ax=Ax,Ay=Ay,Bx=Bx,By=By,alf=alf)
   bd.addL(KL=F0(Ax,Ay,0.04))
   bd.addL(KL=F0(Ax,Ay,0.5*z2*m))
   bd.addL(KL=F0(Bx,By,0.01))
   bd.addL(KL=[(0.0,0.0),(Ax,Ay),(Bx,By)])
   bd.addL(KL=F0(Bx,By,0.04))
   bk.addL(KL=[(Ax,Ay),(Bx,By)])
   bd.add(A=(Ax,Ay),B=(Bx,By))
   fi1+=dfi1
   fi2=(1-z3/z2)*fi1
   if abs(fi1)>=2.0*pi+dfi1: break
bd.addL(KL=F0(0.0,0.0,a))
bd.addL(KL=F0(0.0,0.0,0.5*m*z3))
##================================================== Labels
crl=["red","blue","green","#0DF","#C06","#A64","#686","blue","#AC3"]
bg='White'
Lbd =[Bd(260,355,'ПОЛОЖЕНИЯ',crl[6],bg)]
Lbd+=[Bd(260,380,"yb = [%6.3f  %6.3f]" % (min(bd.L['By']), max(bd.L['By'])),
crl[0],bg)]
Lbd+=[Bd(260,400,"xb = [%6.3f  %6.3f]" % (min(bd.L['Bx']), max(bd.L['Bx'])), crl[1],bg)]
##tt ="fin=%10.3f r\nfik=%10.3f r\n" % (min(g1.bd.L['fi']),max(g1.bd.L['fi']))
##tt+="OA=%10.3f m\nAB=%10.3f m\nBC=%10.3f m\nOC=%10.3f m\n" %
(OA,AB,BC,OC)
##Lbd+=[Bd(410,600,tt,crl[4],bg)]
##================================================== Tkinter
A,B=1000,680
tk=Tk()
tk.title('Кривошип + группа 1-го вида')
fr=Frame(tk)
fr.pack()
c=Canvas(fr,bg=bg,width=A,height=B)
c.pack(expand=1,fill=BOTH)
#================================================== Output
```

```
bg,clr='White',['red','blue','black','green','red','blue','black','green','red','blue',
          "black","green","maroon","#0DF","#C06","#A64","#684","#AC3"]
ww=[4,2,2,1,1,1,1,1,1,1,1,1,1,1,1,1]
ll=[[bd.L['fi1'],bd.L['By']],[bd.L['fi1'],bd.L['Bx']]]
clrd=zip(clr,ww)
desine_d(30,30,300,300,bg,clrd,ll,50,50,c,4,1)
#=========================================================
for ibd in Lbd: ibd.show(c)
mg,xg,yg=0.006,650,280
bk.showL(mg,xg,yg,c,"#A64",4)
for i in ['A']: LayImagine(bd.L[i],mg,xg,yg,c,'red',3)
for i in ['B']: LayImagine(bd.L[i],mg,xg,yg,c,'blue',3)
bd.showL(mg,xg,yg,c,'black',1)

tk.mainloop()
```

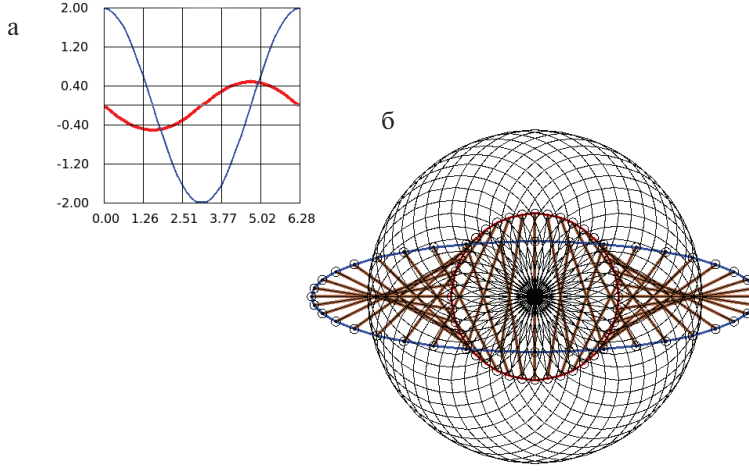
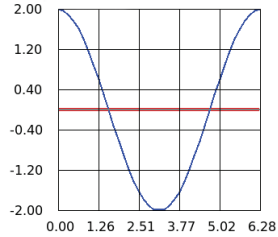


Рис. 7. Планетарно-рычажный механизм с точкой на консоли, длина которой равна длине кривошипа, передаточное число — 2:
а — координаты *x* и *y* точки B на конце консоли по углу поворота кривошипа: б — положения

а



б

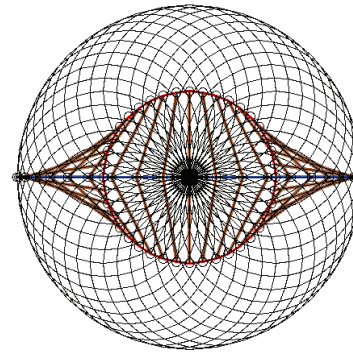
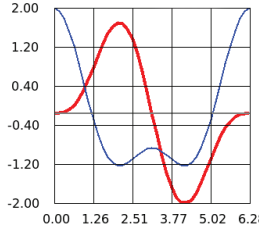


Рис. 8. Планетарно-рычажный механизм с точкой на начальной окружности сателлита, передаточное отношение равно 2:

а — координаты *x* и *y* точки B по углу поворота кривошипа, Y=0.0; б — положения

а



б

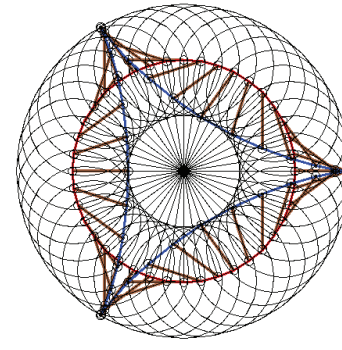


Рис. 9. Планетарно-рычажный механизм с точкой на начальной окружности сателлита, передаточное отношение равно 3:

а — координаты *x* и *y* точки B по углу поворота кривошипа; б — положения

5. Сборка нескольких групп Ассура

Более сложные механизмы синтезируются присоединением новых групп Ассура к уже существующим (рис. 10 на с. 41). В примере к кривошипу сначала присоединена группа первого вида, затем к точке B присоединена группа второго вида и к точке ползуна присоединена группа третьего вида. Метод класса, определяющий положение группы в конкретном положении, выводит сразу все координаты виде кортежа [5].

Здесь длина кривошипа OA=0,25 м; кривошипы AB = 0,9 и BC = 0,8 м; длина OC = 0,7 м; шатун BC = 0,8 м; кулиса DE = 1,0 м.

```
## g123_bsc.py
# -*- coding: utf-8 -*-
from Tkinter import *
from math import *
from bsc_grafik import *
from g1_bsc import *
from g2_bsc import *
from g3_bsc import *
##================================================ Parameter
OA,OC=0.35,0.7
AB,BC=0.9,0.8
```

```
a=-pi*0.2
xc,yc=OC*cos(a),OC*sin(a)
g1=GA1(AB,BC,0.4,0.3,-pi*0.25)
g2=GA2(1.0,0.5)
OC3=0.1
BD31=1.5
BC31=0.10
BS31=0.8
g31=GA31(BD31,BC31,BS31)
xc1=xc*3.0
yc1=yc*1.8
count,fcount=0,4
du,u,uk=pi/36.0,0.0,2.0*pi
while 1:
  xa,ya=OA*cos(u),OA*sin(u)
  xb,yb,xsAB,ysAB,xsBC,ysBC,fiAB,fiBC=g1.position(xa,ya,xc,yc)
  g1.getline(u)
  xb2,yb2,xsAB2,ysAB2,fiAB2=g2.position(xb,yb)
  g2.getline(u)
  g31.position(xb2,yb2,xc1,yc1)
  g31.getline(u)
  if count>=fcount:
    g1.getkontur()
    for i in [g2,g31]: i.getkontur(0)
    count=0
  count+=1
  u+=du
  if u>uk: break
g1.bd.addL(KL=F0(xc,yc,0.01))
g1.bd.addL(KL=F3(xc,yc,0.05,0.1))
g2.bd.addL(KL=F0(0.0,0.0,0.01))
g2.bd.addL(KL=F3(0.0,0.0,0.05,0.1))
g31.bd.addL(KL=F0(xc1,yc1,0.01))
g31.bd.addL(KL=F3(xc1,yc1,0.05,0.1))
##================================================ Labels
crl=["black","red","blue","green","#0DF","#C06","#A64","#686","blue","#AC3"]
bg='White'
Lbd =[Bd(500,55,'ПЛАН ПОЛОЖЕНИЙ',crl[6],bg)]
Lbd+=[Bd(125, 325,"ys3 = [%6.3f  %6.3f]" % (min(g31.bd.L['ys1']), max(g31.
bd.L['ys1'])), crl[2],bg)]
Lbd+=[Bd(125, 345,"xs3 = [%6.3f  %6.3f]" % (min(g31.bd.L['xs1']), max(g31.
```

```
bd.L['xs1'])), crl[1],bg)]
Lbd+=[Bd(125, 425,"yb = [%6.3f  %6.3f]" % (min(g1.bd.L['yb']), max(g1.bd.L['yb'])),
crl[2],bg)]
Lbd+=[Bd(125, 445,"xb = [%6.3f  %6.3f]" % (min(g1.bd.L['xb']), max(g1.bd.L['xb'])),
crl[1],bg)]
tt ="fin=%10.3f r\nfik=%10.3f r\n" % (min(g2.bd.L['fi']),max(g2.bd.L['fi']))
tt+="OA=%10.3f m\nAB=%10.3f m\nBC=%10.3f m\nOC=%10.3f m\n" %
(OA,AB,BC,OC)
Lbd+=[Bd(440,480,tt,crl[0],bg)]
tt="xa = [%6.3f   %6.3f]\n" % (min(g2.bd.L['xa']), max(g2.bd.L['xa']))
tt+="ya = [%6.3f   %6.3f]\n" % (min(g2.bd.L['ya']), max(g2.bd.L['ya']))
tt+="xb = [%6.3f   %6.3f]\n" % (min(g2.bd.L['xb']), max(g2.bd.L['xb']))
tt+="yb = [%6.3f   %6.3f]\n" % (min(g2.bd.L['yb']), max(g2.bd.L['yb']))
Lbd+=[Bd(660,480,tt,crl[0],bg)]
Lbd+=[Bd(400,150,'A',crl[0],bg)]
Lbd+=[Bd(550,130,'B',crl[0],bg)]
Lbd+=[Bd(950,260,'S3',crl[0],bg)]
##================================================== Tkinter
A,B=1000,680
tk=Tk()
tk.title('Кривошип + группы 1-, 2- и 3-го вида')
fr=Frame(tk)
fr.pack()
c=Canvas(fr,bg=bg,width=A,height=B)
c.pack(expand=1,fill=BOTH)
#==================================================== Output
bg,clr='White',['red','blue','black','green','red','blue','black','green','red','blue',]
ww=[4,2,2,1,1,1,1,1,1,1,1,1,1,1,1,1,1]
clrd=zip(clr,ww)
ll=[[g1.bd.L['fi'],g1.bd.L['yb']],[g1.bd.L['fi'],g1.bd.L['xb']]]
desine_d(20,360,300,300,bg,clrd,ll,50,50,c,4,1)
ll=[[g1.bd.L['fi'],g31.bd.L['ys1']],[g1.bd.L['fi'],g31.bd.L['xs1']]]
desine_d(20,10,300,300,bg,clrd,ll,50,50,c,4,1)
#=====================================================
for ibd in Lbd: ibd.show(c)
mg,xg,yg=0.004,420,250
g1.putline(c,mg,xg,yg,['A','B','C'],'blue',3)
g1.putline(c,mg,xg,yg,['S1','S2'],'green',3)
g2.putline(c,mg,xg,yg,['B'],'blue',3)
g31.putline(c,mg,xg,yg,['S1'],'green',3)
g31.putline(c,mg,xg,yg,['D'],'blue',3)
```

```
for i in [g1,g2,g31]: i.putkontur(c,mg,xg,yg)
tk.mainloop()
```

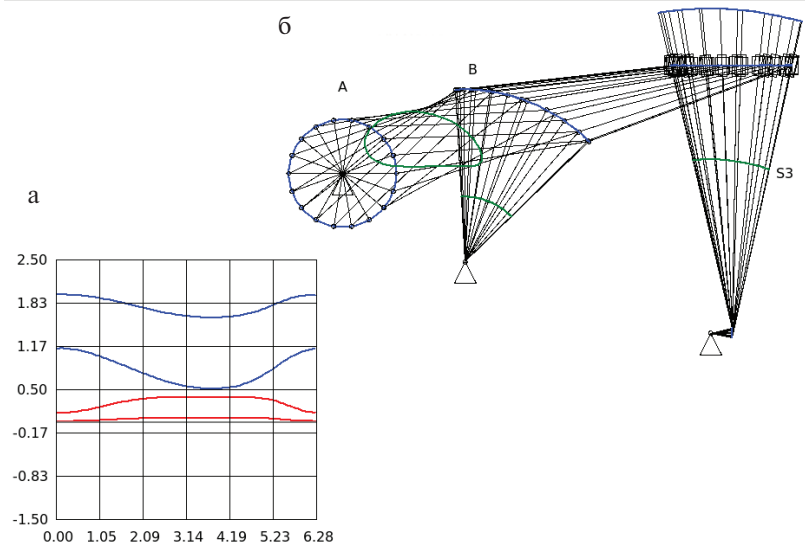


Рис. 10 Рычажный механизм, собранный
из первых трех групп Ассура:

а — координаты *x* и *y* точки B по углу поворота кривошипа;
б — план положений

6. Примеры исследования механизмов

С помощью языка Python исследуем механизмы с группой Ассура первого вида (рис. 11–13 на с. 46–47). Для разных операций используем разные страницы. Длины звеньев такие же, как в пункте 3.1.

```
## g1_bsc_proj.py
# -*- coding: utf-8 -*-
from Tkinter import *
from bsc_grafik import *
from g1_bsc import *
from math import *
from numpy import *
from numpy import linalg

##========================================== Объект 1-го вида
def show():
    OA,OC=0.4,1.0
    AB,BC=0.9,0.8
    AS,CQ=0.5,0.45
    a=-pi*0.35
    xc,yc=OC*cos(a),OC*sin(a)
    g1=GA1(AB,BC,AS,CQ,a)
```

```
  mAB,mBC,JAB,JBC=1.0,1.0,1.0,1.0
  w,Eps,g=1.0,0.0,9.81
  P1,P2=mAB*g,mBC*g
  count,fcount=0,4
  du=pi/36.0
  u=0.0
  uk=2.0*pi
  bd=Bl('fi','dxb','dyb','ddxb','ddyb','JpAB','JpBC','Jp','RxB','RyB')
  while 1:
    xa,ya=OA*cos(u),OA*sin(u)
    xa2,ya2=OA*cos(u+0.01),OA*sin(u+0.01)
    xa1,ya1=OA*cos(u-0.01),OA*sin(u-0.01)
    xb,yb,xsAB,ysAB,xsBC,ysBC,fiAB,fiBC=g1.position(xa,ya,xc,yc)
    xb2,yb2,xsAB2,ysAB2,xsBC2,ysBC2,fiAB2,fiBC2=g1.position(xa2,ya2,xc,yc)
    xb1,yb1,xsAB1,ysAB1,xsBC1,ysBC1,fiAB1,fiBC1=g1.position(xa1,ya1,xc,yc)

    dxb,dyb,dxsAB,dysAB=(xb2-xb1)/0.02,(yb2-yb1)/0.02,(xsAB2-xsAB1)/0.02,
                        (ysAB2-ysAB1)/0.02
    dxsBC,dysBC,dfiAB,dfiBC=(xsBC2-xsBC1)/0.02,(ysBC2-ysBC1)/0.02,
                            (fiAB2-fiAB1)/0.02,(fiBC2-fiBC1)/0.02
    ddxb,ddyb=(xb2-2.0*xb+xb1)/0.02/0.02,(yb2-2.0*yb+yb1)/0.02/0.02
    ddxsAB,ddysAB=(xsAB2-2.0*xsAB+xsAB1)/0.02/0.02,
                  (ysAB2-2.0*ysAB+ysAB1)/0.02/0.02
    ddxsBC,ddysBC=(xsBC2-2.0*xsBC+xsBC1)/0.02/0.02,(ysBC2-2.0*ysBC+ysBC1)/0.02/0.02
    ddfiAB,ddfiBC=(fiAB2-2.0*fiAB+fiAB1)/0.02/0.02,(fiBC2-
2.0*fiBC+fiBC1)/0.02/0.02
    JpAB=mAB*dxsAB**2+mAB*dysAB**2+JAB*dfiAB**2
    JpBC=mBC*dxsBC**2+mBC*dysBC**2+JBC*dfiBC**2
    Jp=JpAB+JpBC
  ##====================================
    F1x=dxsAB*Eps+ddxsAB*w**2
    F2x=dxsBC*Eps+ddxsBC*w**2
    F1y=dysAB*Eps+ddysAB*w**2
    F2y=dysBC*Eps+ddysBC*w**2
    M1=dfiAB*Eps+ddfiAB*w**2
    M2=dfiBC*Eps+ddfiBC*w**2

    hx1,hy1,hx2,hy2,lx1,ly1,lx2,ly2=xsAB-xa,ysAB-yb,xsBC-xb,ysBC-yb,xb-xsAB,yb-ysAB,
                                    xc-xsBC,yc-ysBC
    ##   RxA  RyA  RxB  RyB  RxC  RyC
    ma=[[ hy1,-hx1,-ly1, lx1, 0.0, 0.0],##AB:M
```

```
        [ 1.0, 0.0, 1.0, 0.0, 0.0, 0.0],##X
        [ 0.0, 1.0, 0.0, 1.0, 0.0, 0.0],##Y
        [ 0.0, 0.0, hy2, hx2, ly2,-lx2],##BC:M
        [ 0.0, 0.0,-1.0, 0.0, 1.0, 0.0],##X
        [ 0.0, 0.0, 0.0,-1.0, 0.0, 1.0]]##Y
      mb=[-M1, -F1x, -F1y+P1, -M2, -F2x, -F2y+P2]
      Ma,Mb = array((ma)),array((mb))
      Mx = solve_linear_equations(Ma,Mb)
  ##====================================
bd.add(fi=u,dxb=dxb,dyb=dyb,ddxb=ddxb,ddyb=ddyb,JpAB=JpAB,JpBC=JpBC,Jp=Jp,
        RxB=Mx[2],RyB=Mx[3])
      g1.getline(u)
      if count>=fcount:
        g1.getkontur()
        count=0
      count+=1
      u+=du
      if u>uk: break
  g1.bd.addL(KL=F0(xc,yc,0.01))
  g1.bd.addL(KL=F3(xc,yc,0.05,0.1))
  g1.bd.addL(KL=F0(0.0,0.0,0.01))
  g1.bd.addL(KL=F3(0.0,0.0,0.05,0.1))
  ##=================================================  Labels
  bg,clr='White',['red','blue','green','black','maroon','#0DF','#C06','#A64','#684','#AC3']
  ww=[4,4,4,4,4,4,4,4,4,1,1,1,1,1,1,1,1]
  clrd=zip(clr,ww)
  Lbd =[Bd(700,55,'ПЛАН ПОЛОЖЕНИЙ',clr[3],bg)]
  Lbd+=[Bd(95, 350,"yb = [%6.3f  %6.3f]" % (min(g1.bd.L['yb']), max(g1.bd.L['yb'])),
'blue',bg)]
  Lbd+=[Bd(95, 370,"xb = [%6.3f  %6.3f]" % (min(g1.bd.L['xb']), max(g1.bd.L['xb'])),
'red',bg)]
  Lbd+=[Bd(95, 390,"fi = [%6.3f  %6.3f]" % (min(g1.bd.L['fi']), max(g1.bd.L['fi'])),
'black',bg)]
  tt ="OA=%10.3f m\nAB=%10.3f m\nBC=%10.3f m\nOC=%10.3f m\n" %
(OA,AB,BC,OC)
  tt+="CQ=%10.3f m\nalfa=%10.3f рад\n" % (CQ,a)
  Lbd+=[Bd(110,500,tt,'black',bg)]

  ## Положение
  A,B=1000,650
  tk=Tk()
```

```
  tk.title('Кривошип + группа 1-го вида')
0   fr=Frame(tk)
  fr.pack()
  c=Canvas(fr,bg=bg,width=A,height=B)
  c.pack(expand=1,fill=BOTH)
  mg=0.0025
  xg,yg=500,200
  for ibd in Lbd: ibd.show(c)
  g1.putline(c,mg,xg,yg,['A','B','C'],'red',3)
  g1.putline(c,mg,xg,yg,['S1','S2'],'green',5)
  g1.putkontur(c,mg,xg,yg)
  ll=[[g1.bd.L['fi'],g1.bd.L['yb']],[g1.bd.L['fi'],g1.bd.L['xb']]]
  desine_d(30,30,300,300,bg,clrd,ll,50,50,c,4,1)

  ## Приведение
  tk1=Tk()
  tk1.title('Силовой расчет')
  fr=Frame(tk1)
  fr.pack()
  c=Canvas(fr,bg=bg,width=A,height=B)
  c.pack(expand=1,fill=BOTH)
  ll=[[bd.L['fi'],bd.L['RxB']]]
  desine_d(30,20,600,300,bg,clrd[0:],ll,50,50,c,4,1)
  ll=[[bd.L['fi'],bd.L['RyB']]]
  desine_d(30,340,600,300,bg,clrd[1:],ll,50,50,c,4,1)
  ll=[[bd.L['RxB'],bd.L['RyB']]]
  desine_d(660,20,300,300,bg,clrd[2:],ll,50,50,c,4,1)

  ## Силовой расчет
  tk2=Tk()
  tk2.title('Приведение параметров')
  fr=Frame(tk2)
  fr.pack()
  c=Canvas(fr,bg=bg,width=A,height=B)
  c.pack(expand=1,fill=BOTH)
  ll=[[g1.bd.L['fi'],g1.bd.L['xb']],[bd.L['fi'],bd.L['dxb']]]
  desine_d(30,20,950,300,bg,clrd[0:],ll,50,50,c,4,1)
  ll=[[bd.L['fi'],bd.L['ddxb']]]
  desine_d(30,20,950,300,bg,clrd[2:],ll,50,50,c,4,1)
  ll=[[bd.L['fi'],bd.L['JpAB']],[bd.L['fi'],bd.L['JpBC']],[bd.L['fi'],bd.L['Jp']]]
  desine_d(30,340,950,300,bg,clrd[4:],ll,50,50,c,4,1)
```

```
    tt="CQ=%10.3f m\nalfa=%10.3f рад" % (CQ,a)
    Lbd=[Bd(210,500,tt,'black',bg)]
    for ibd in Lbd: ibd.show(c)
    ##-------------
    tk2.mainloop()
    tk1.mainloop()
    tk.mainloop()
if __name__=="__main__": show()
```

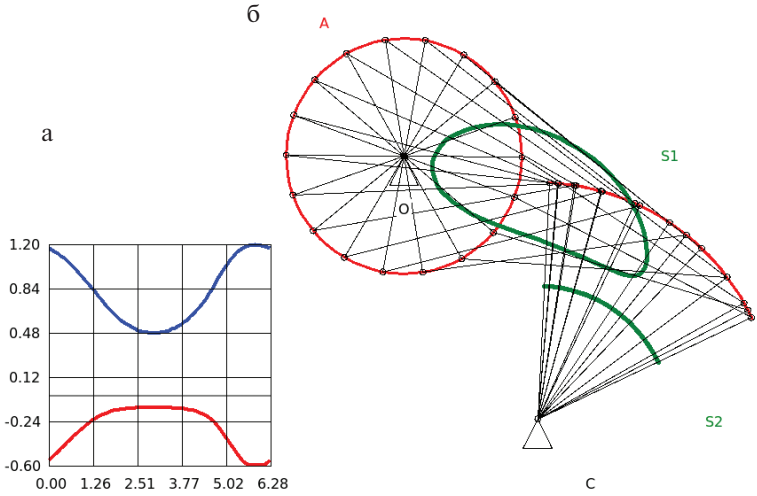


Рис. 11. Кинематика группы первого вида:
а — координаты *x* и *y* точки B по углу поворота кривошипа;
б — план положений

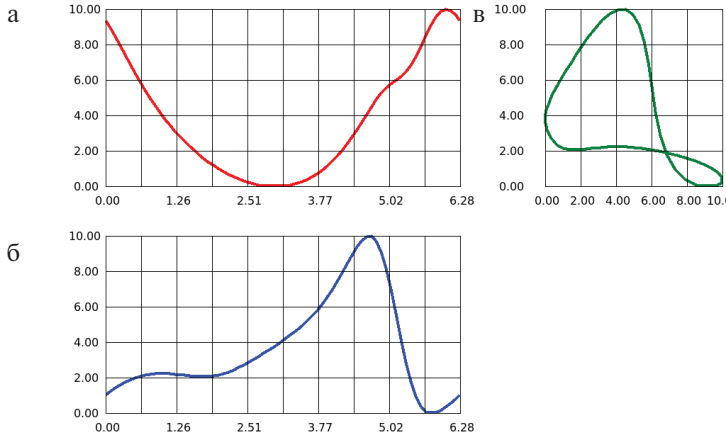


Рис. 12. Силовой расчет группы первого вида:

а — координата *x* силы, действующей в точке B по углу поворота кривошипа;
б — координата *y* силы, действующей в точке B по углу поворота кривошипа;
в — годограф силы, действующей в точке B по углу поворота кривошипа

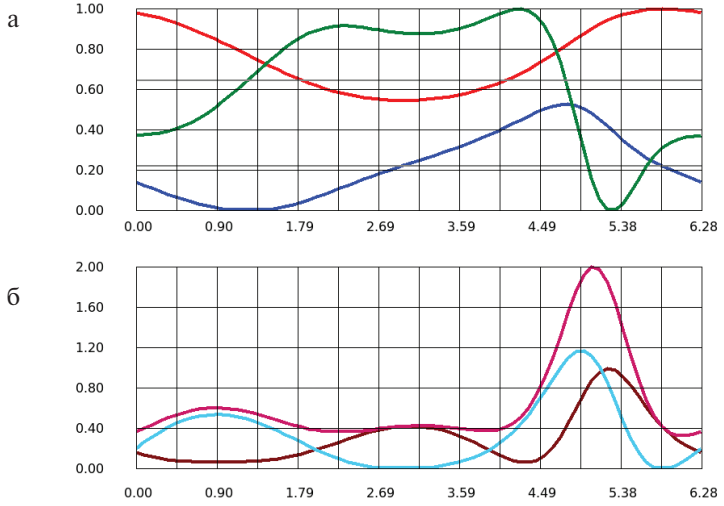


Рис. 13. Динамический расчет группы Ассура первого вида:

а — аналоги ускорения точки B и угла поворота звена AB по углу поворота кривошипа; б — приведенные моменты инерции звеньев AB и BC и их суммарный момент

Приведем пример исследования механизма с группой Ассура второго вида (рис. 14–16 на с. 52–53). Здесь тот же набор операций, что и в предыдущем случае.

Длины звеньев такие же, как в пункте 3.2.

```
## g2_bsc_proj.py
# -*- coding: utf-8 -*-
from Tkinter import *
from math import *
from numpy import *
from numpy import linalg
from bsc_grafik import *
from g2_bsc import *
def show():
  OA,AB,AS=0.25,0.9,0.4
  a,e=0.0,0.02
  g2=GA2(AB,e,a,AS)
  JAB,mAB,mB=5.0,2.0,10.0
  w,g=1.0,9.81
  P1,P2=mAB*g,mB*g
  count,fcount=0,4
  du,u,uk=pi/36.0,0.0,2.0*pi
  bd=Bl('fi','dxb','dyb','ddxb','ddyb','JpAB','JpB','Jp','RxA','RyA','RxB','RyB','N','xsAB','ysAB',
      'dxsAB','dysAB','ddxsAB','ddysAB','F1x','F1y','F2x','F2y')
  while 1:
    xa,ya=OA*cos(u),OA*sin(u)
    xb, yb, xsAB, ysAB, fiAB =g2.position(xa, ya)
    xb2,yb2,xsAB2,ysAB2,fiAB2=g2.position(OA*cos(u+0.01),OA*sin(u+0.01))
    xb1,yb1,xsAB1,ysAB1,fiAB1=g2.position(OA*cos(u-0.01),OA*sin(u-0.01))
    dxb=(xb2-xb1)/0.02
    dyb=(yb2-yb1)/0.02
    dxsAB=(xsAB2-xsAB1)/0.02
    dysAB=(ysAB2-ysAB1)/0.02
    dfiAB=(fiAB2-fiAB1)/0.02
    ddxb=(xb2-2.0*xb+xb1)/0.0004
    ddyb=(yb2-2.0*yb+yb1)/0.0004
    ddxsAB=(xsAB2-2.0*xsAB+xsAB1)/0.0004
```

```
    ddysAB=(ysAB2-2.0*ysAB+ysAB1)/0.0004
    ddfiAB=(fiAB2-2.0*fiAB+fiAB1)/0.0004
    JpAB=mAB*dxsAB**2+mAB*dysAB**2+JAB*dfiAB**2
    JpB=mB*dxb**2+mB*dyb**2
    Jp=JpAB+JpB
  ##====================================
    F1x=mAB*ddxsAB*w**2
    F2x= mB*ddxb*w**2
    F1y=mAB*ddysAB*w**2
    F2y= mB*ddyb*w**2
    M1= JAB*ddfiAB*w**2
    xN,yN=cos(a),sin(a)
    hx,hy,lx,ly=AS*sin(u),AS*cos(u),(AB-AS)*sin(u),(AB-AS)*cos(u)
    ##   RxA  RyA  RxB  RyB   N
    ma=[[ -hy,  hx,  ly, -lx, 0.0],##AB:M
       [ 1.0, 0.0, 1.0, 0.0, 0.0],##X
       [ 0.0, 1.0, 0.0, 1.0, 0.0],##Y
       [ 0.0, 0.0,-1.0, 0.0,  xN],##B:X
       [ 0.0, 0.0, 0.0,-1.0,  yN]]##Y
    mb=[-M1, -F1x, -F1y+P1, -F2x, -F2y+P2]
    Ma,Mb = array((ma)),array((mb))
    Mx = solve_linear_equations(Ma,Mb)
  ##====================================
bd.add(fi=u,dxb=dxb,dyb=dyb,ddxb=ddxb,ddyb=ddyb,JpAB=JpAB,JpB=JpB,Jp=Jp,
       RxA=Mx[0],RyA=Mx[1],RxB=Mx[2],RyB=Mx[3],N=Mx[4],
xsAB=xsAB,ysAB=ysAB,dxsAB=dxsAB,dysAB=dysAB,ddxsAB=ddxsAB,ddysAB=ddysAB,
        F1x=F1x,F1y=F1y,F2x=F2x,F2y=F2y)
    g2.getline(u)
    if count>=fcount:
       g2.getkontur()
       count=0
    count+=1
    u+=du
    if u>uk: break
  g2.bd.addL(KL=F0(0.0,0.0,0.01))
  g2.bd.addL(KL=F3(0.0,0.0,0.05,0.1))
  ##================================================ Labels
  crl=["black","red","green","#0DF","#C06","#A64","#686","blue","#AC3"]
  bg='White'
  Lbd =[Bd(600,55,'ПЛАН ПОЛОЖЕНИЙ',crl[6],bg)]
  Lbd+=[Bd(120, 370,"xb = [%6.3f  %6.3f]" % (min(g2.bd.L['xb']),max(g2.
bd.L['xb'])),'red',bg)]
```

```
Lbd+=[Bd(120, 390,"fi = [%6.3f  %6.3f]" % (min(g2.bd.L['fi']), max(g2.bd.L['fi'])),'black',bg)]
tt ="OA=%10.3f m\nAB=%10.3f m\n" % (OA,AB)
tt+="e=%10.3f m\nAS=%10.3f m\n" % (e,AS)
tt+="alfa=%10.3f m\n" % (a)
Lbd+=[Bd(720,360,tt,"black",bg)]
##============================================== Кинематика
A,B=1000,680
tk=Tk()
tk.title('Кривошип + группа 2-го вида')
fr=Frame(tk)
fr.pack()
c=Canvas(fr,bg=bg,width=1000,height=450)
c.pack(expand=1,fill=BOTH)
bg,clr='White',['red','blue',"black",'green','red','blue',"black",'green','red','blue',
          "black",'green','maroon',"#0DF","#C06","#A64","#684","#AC3"]
ww=[4,2,2]
clrd=zip(clr,ww)
ll=[[g2.bd.L['fi'],g2.bd.L['xb']]]
desine_d(30,30,300,300,bg,clrd,ll,50,50,c,4,1)
mg,xg,yg=0.0023,450,200
for ibd in Lbd: ibd.show(c)
g2.putline(c,mg,xg,yg,['S1'],'green',3)
g2.putline(c,mg,xg,yg,['A','B'],'blue',3)
g2.putkontur(c,mg,xg,yg)
Lbd=[Bd(410,80,'A','blue',bg)]
Lbd+=[Bd(910,160,'B','blue',bg)]
Lbd+=[Bd(680,120,'S1','green',bg)]
Lbd+=[Bd(450,260,'O','black',bg)]
for ibd in Lbd: ibd.show(c)
#=================================== Приведенные параметры
tk1=Tk()
tk1.title('Приведенные параметры')
fr=Frame(tk1)
fr.pack()
c=Canvas(fr,bg=bg,width=A,height=B)
c.pack(expand=1,fill=BOTH)
ll=[[bd.L['fi'],bd.L['dxb']],[bd.L['fi'],bd.L['ddxb']]]
desine_d(30,30,600,300,bg,clrd,ll,50,50,c,4,1)
ll=[[bd.L['fi'],bd.L['Jp']],[bd.L['fi'],bd.L['JpB']],[bd.L['fi'],bd.L['JpAB']]]
desine_d(30,350,600,300,bg,clrd,ll,50,50,c,4,1)
sf='%s= [%8.3f  %8.3f] %s'
tt =sf % ('dxb',min(bd.L['dxb']),max(bd.L['dxb']),'m/rad\n')
```

```
    tt+=sf % ('ddxb',min(bd.L['ddxb']),max(bd.L['ddxb']),'m/rad/rad\n')
    tt+=sf % ('JpAB',min(bd.L['JpAB']),max(bd.L['JpAB']),'kg*m*m\n')
    tt+=sf % ('JpB',min(bd.L['JpB']),max(bd.L['JpB']),'kg*m*m\n')
    tt+=sf % ('Jp',min(bd.L['Jp']),max(bd.L['Jp']),'kg*m*m\n')
    tt+=sf % ('fi',min(bd.L['fi']),max(bd.L['fi']),'rad')
    Lbd=[Bd(820,250,tt,'black',bg)]
    Lbd+=[Bd(70,310,'dxb(fi)','red',bg)]
    Lbd+=[Bd(70,160,'ddxb(fi)','blue',bg)]
    Lbd+=[Bd(55,535,'JpAB','black',bg)]
    Lbd+=[Bd(55,580,'JpB','blue',bg)]
    Lbd+=[Bd(60,380,'Jp(fi)','red',bg)]
    for ibd in Lbd: ibd.show(c)
    #========================================== Силовой расчет
    tk2=Tk()
    tk2.title('Силовой расчет')
    fr=Frame(tk2)
    fr.pack()
    c=Canvas(fr,bg=bg,width=A,height=B)
    c.pack(expand=1,fill=BOTH)
    ll=[[bd.L['fi'],bd.L['RyA']]]
    desine_d(30,30,600,300,bg,clrd,ll,50,50,c,4,1)
    ll=[[bd.L['fi'],bd.L['F1x']],[bd.L['fi'],bd.L['F1y']]]
    desine_d(30,350,600,300,bg,clrd,ll,50,50,c,4,1)
    ll=[[bd.L['F1x'],bd.L['F1y']]]
    desine_d(660,30,300,300,bg,clrd,ll,50,50,c,4,1)
    tt =sf % ('RyA',min(bd.L['RyA']),max(bd.L['RyA']),'N\n')
    tt+=sf % ('F1x',min(bd.L['F1x']),max(bd.L['F1y']),'N\n')
    tt+=sf % ('fi',min(bd.L['fi']),max(bd.L['fi']),'rad')
    Lbd=[Bd(820,500,tt,'black',bg)]
    Lbd+=[Bd(70,580,'F1x(fi)','red',bg)]
    Lbd+=[Bd(70,455,'F1y(fi)','blue',bg)]
    Lbd+=[Bd(710,155,'F1x(F1y)','red',bg)]
    Lbd+=[Bd(65,50,'RyA(fi)','red',bg)]
    for ibd in Lbd: ibd.show(c)
    #=========================================================
    tk2.mainloop()
    tk1.mainloop()
    tk.mainloop()
if __name__=="__main__": show()
```

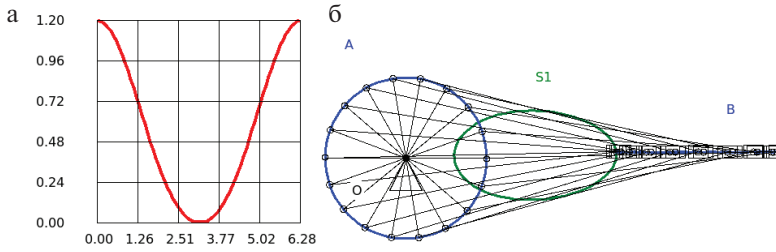


Рис. 14. Кинематический расчет группы второго вида:
а– координаты *x* и *y* точки B по углу поворота кривошипа;
б — план положений

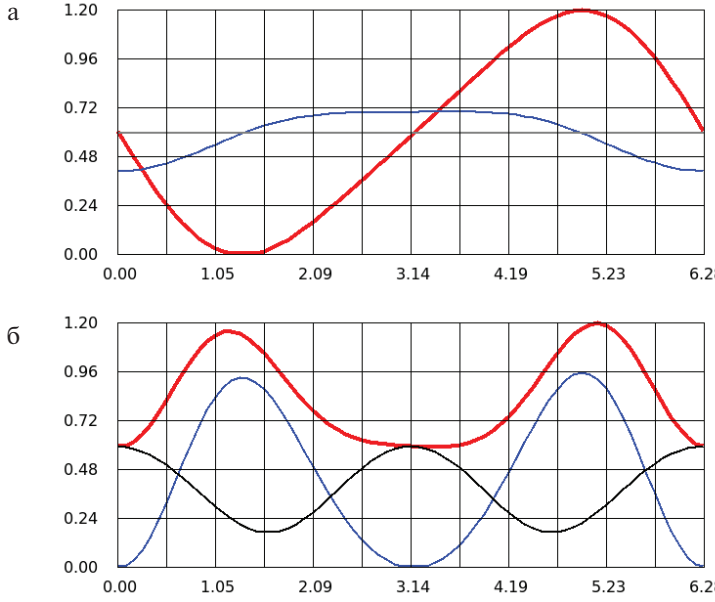


Рис. 15. Динамический расчет группы второго вида:
а — аналоги ускорений координат *x* и *y* точки B по углу поворота кривошипа;
б — приведенные моменты инерции и суммарный момент массы, сосредоточенной в точке B с координатами *x* и *y* по углу поворота кривошипа

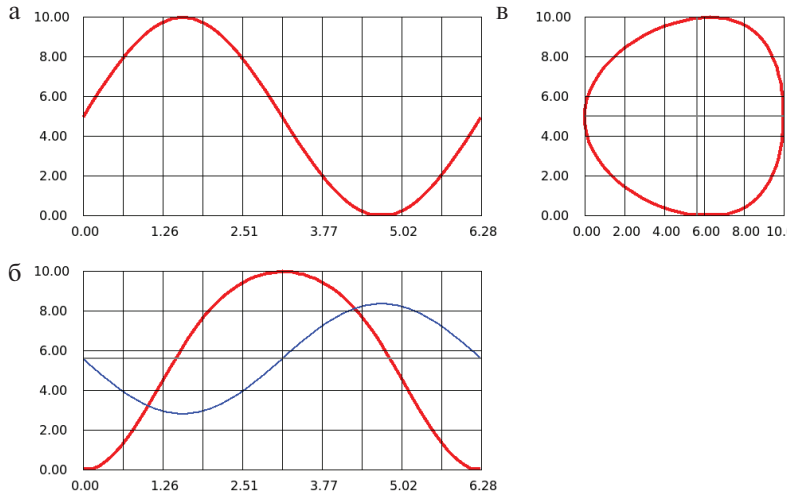


Рис. 16. Силовой расчет группы Ассура второго вида:

а — координата *y* вектора силы действующей в точки В по углу поворота кривошипа; б — координата *x* вектора силы, действующей в точке В, и приведенная к углу кривошипа сила инерции, действующей по координате *y*; в — траектория конца вектора силы, действующей в точке В

7. Влияние двигателя

Момент двигателя с ограниченной мощностью реагирует на изменение параметров кинематической цепи. Переменный характер момента приводит к колебаниям скорости двигателя и ведущего звена. Параметры единичного электродвигателя после задания момента и скорости автоматически приводятся к координате поворота кривошипа (рис. 17 на с. 56).

```
## dvig_0_bsc.py
# -*- coding: utf-8 -*-
from math import *
from bsc_grafik import *
from bsc_dinam import *
from Tkinter import *
#=====================================================
Jd = M(30.0)
kw,km=6.0,2000.0
dvig=Dvig(Jd,kw,km)
V=[Jd,dvig]
E=[dvig]
bt=Bl('t')
t, dt, tkon = 0.0, 0.0005, 5.0
while t<tkon:
   for i in V: i.f=0.0
   for j in E: j.F()
   for i in V: i.step(dt)
```

```
  if t>=0.0:
    for i in V+E: i.add()
    bt.add(t=t)
  t+=dt
##================================================== Tkinter
A,B = 800,330
tk = Tk()
tk.title("Новая трансмиссия")
f = Frame(tk)
f.pack()
c = Canvas(f, bg="white", width=A, height=B)
c.pack(expand=1, fill=BOTH)
bg,clr='White',['red','blue','black','green','maroon',"#0DF","#C06","#A64","#684","#AC3"]
ww=[2,2,2,1,2,4,2,2,2,2]
##================================================== Output
clrd=zip(clr,ww)
ll=[[dvig.bd.L['f'],dvig.bd.L['v']]]
desine_d(520,10,250,250,bg,[("green",2)],ll,50,50,c,1,1)
#=========================================================
LL=[Jd.bd.L['v']]
desine_l(10,10,500,250,bg,clrd[0:],bt.L['t'],LL,0,50,50,c,1,1)
LL=[dvig.bd.L['f']]
desine_l(10,10,500,250,bg,[('blue',3)],bt.L['t'],LL,0,50,50,c,1,1)
#=========================================================
Lbd =[Bd(130,45,'w(t)','red',bg)]
Lbd+=[Bd(130,225,'M(t)','blue',bg)]
Lbd+=[Bd(595,225,'w(M)','green',bg)]
sf="%3s = %9.3f %5s"
tt =sf % ('t',t,'c\n')
tt+=sf % ('w',max(dvig.bd.L['v']),'1/c\n')
tt+=sf % ('M',max(dvig.bd.L['f']),'Mn')
Lbd+=[Bd(150,295,tt,'black',bg)]
for ibd in Lbd: ibd.show(c)
tk.mainloop()
```

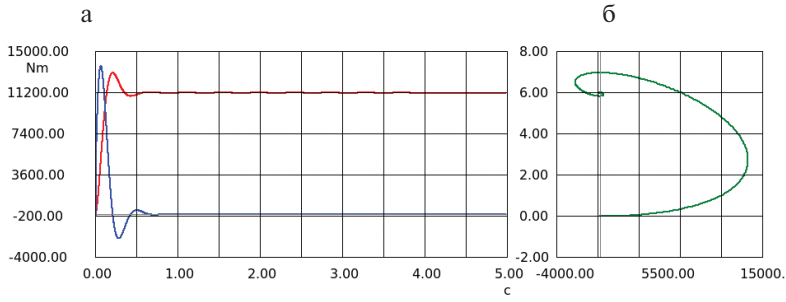


Рис. 17. Движение электродвигателя:

а — момент $M(t)$ и скорость ротора двигателя w(t) по времени;
б — фазовый портрет $w(M)$ двигателя

8. Влияние приведенных к кривошипу сил и масс

При движении кинематической цепи меняются передаточные функции, а значит меняются приведенный момент внешних сил и момент инерции. Это приводит к изменению скорости и момента двигателя. Кривошип, ведомый электродвигателем, соединен с группой Ассура второго вида (рис. 18 на с. 60).

```
## dvig_3R_bsc.py
# -*- coding: utf-8 -*-
from Tkinter import *
import math
from bsc_grafik import *
from bsc_dinam import *

class ST_KL:   ##ползун
   def __init__(self,m=1.0,r1=0.5,r2=1.75,d=0.15):
      self.m=m
      self.r1,self.r2,self.d=r1,r2,d
   def f(self,u):
      r1,r2,d=self.r1,self.r2,self.d
      ay=r1*sin(u)
```

```
      ax=r1*cos(u)
      gamma=asin((ay-d)/r2)
      bx=ax+r2*cos(gamma)
      return bx
   def df(self,u): return (self.f(u+0.01)-self.f(u-0.01))/0.02
   def ddf(self,u):return (self.df(u+0.01)-self.df(u-0.01))/0.02
   def J(self,u):  return self.m*self.df(u)**2
   def dJ(self,u): return 2.0*self.m*self.ddf(u)*self.df(u)

##=======================================================
J=250.0
Jr=M(J)
kw,km=10.0,25000.0
dvig=Dvig(Jr,kw,km)
mkl=1450.0
r1=0.25
r2=1.75
r4=0.5
d=0.15
#------------------------
Kl=ST_KL(mkl,r1,r2,d)
V=[Jr,dvig]
E=[dvig]
pi2=pi*2.0
t, dt, tkon = 0.0, 0.001, 1.50
bd=Bl('t','Mr','wr','Jr','x','dx','Mc','Mk','My')
while t<tkon:
   for i in V: i.f=0.0
   fi=Jr.x[0]
   u=fmod(fi,pi2)
   Jr.m=J+Kl.J(u)
   xx=Kl.f(u)
   pfk=Kl.df(u)
   Mk=0.5*(Kl.dJ(u))*Jr.x[1]**2
   Mo=-mkl*r4*cos(u)
   Jr.f=Mk+Mo
   for j in E: j.F()
   for i in V: i.step(dt)
   bd.add(t=t,Mr=Jr.f,wr=Jr.x[1],x=xx,dx=pfk,Mk=Mk)
   if t>=0.0:
      for i in V+E: i.add()
```

```
   t+=dt
#=========================================================
A,B=1000,460
tk = Tk()
tk.title("Одномассовый машинный агрегат")
f = Frame(tk)
f.pack()
c = Canvas(f, bg="White", width=A, height=B)
c.pack(expand=1, fill=BOTH)
#=========================================================
clr=['blue','black','red','green','blue','gray','"#C06"','"#A64"','"#684"','"#AC3"','"#0DF"','blue','green']
ww=(2,2,2,2,1,3,3,1,1,1,1,1,1,1)
cl=zip(clr,ww)
bg="White"
mg,xg,yg=0.004,400,200
desine_l(650,30, 300,300,(255,255,255),cl[0:],bd.L['Mr'],[bd.L['wr']],0,50,50,c,1,1)
LL=[bd.L['Mr']]
desine_l(20, 30, 600,300,(255,255,255),cl[1:],bd.L['t'], LL,0,50,50,c,1,1)
desine_l(20, 30, 600,300,(255,255,255),cl[3:],bd.L['t'], [bd.L['wr']],0,50,50,c,1,1)
ff2,tt='%s %15.5f %15.5f %s','
ff1='%15s %15.5f %s'
ff0='%15s %15.5f'
Lbd =[Bd(400,100,'w(t)','green',bg)]
Lbd+=[Bd(400,300,'M(t)','black',bg)]
Lbd+=[Bd(825,300,'w(M)','blue',bg)]
tt ='ПАРАМЕТРЫ\n'
tt+=ff1 % ('kw=',kw,'1/c\n')
tt+=ff1 % ('km=',km,'Нм')
Lbd+=[Bd(800,400,tt,'black',bg)]
tt ='РЕЗУЛЬТАТЫ\n'
tt+=ff2 % ('t =',min(bd.L['t']),max(bd.L['t']),'c \n')
tt+=ff2 % ('Wr=',min(bd.L['wr']),max(bd.L['wr']),'1/c \n')
tt+=ff2 % ('Mr=',min(bd.L['Mr']),max(bd.L['Mr']),'Нм \n')
tt+=ff2 % ('Mk=',min(bd.L['Mk']),max(bd.L['Mk']),'Нм \n')
Lbd+=[Bd(200,400,tt,'black',bg)]
for ibd in Lbd: ibd.show(c)
tk.mainloop()
```

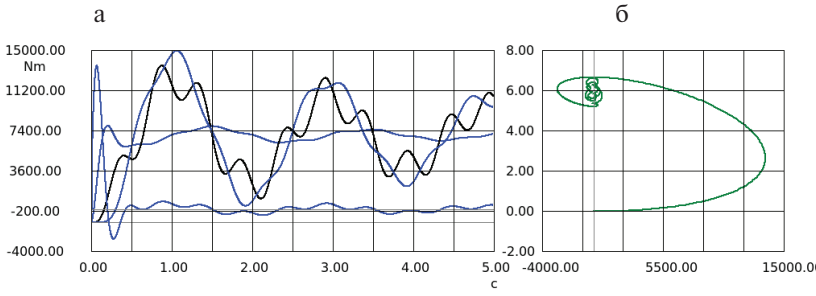


Рис. 18. Движение машинного агрегата
с жесткой кинематической цепью и электродвигателем:
а — момент $M(t)$ и скорость $w(t)$ ротора двигателя по времени;
б — фазовый портрет $w(M)$ двигателя

9. Влияние упругих элементов

Движение машинного агрегата с кинематической цепью деформируют упругие элементы (звенья). Это приводит к действию переменных упругих сил. Во вращательные пары группы вставлены упругие элементы (рис. 19 на с. 64).

```
## dvig_3C_bsc.py
# -*- coding: utf-8 -*-
from Tkinter import *
import math
from bsc_grafik import *
from bsc_dinam import *
class ST_KL:    ##ползун
   def __init__(self,m=1.0,r1=0.5,r2=1.75,d=0.15):
      self.m=m
      self.r1,self.r2,self.d=r1,r2,d
   def f(self,u):
      r1,r2,d=self.r1,self.r2,self.d
      ay=r1*sin(u)
      ax=r1*cos(u)
      gamma=asin((ay-d)/r2)
      bx=ax+r2*cos(gamma)
```

```
      return bx
   def df(self,u): return (self.f(u+0.01)-self.f(u-0.01))/0.02
   def ddf(self,u):return (self.df(u+0.01)-self.df(u-0.01))/0.02
   def J(self,u):  return self.m*self.df(u)**2
   def dJ(self,u): return 2.0*self.m*self.ddf(u)*self.df(u)
##=======================================================
J=250.0
Jr=M(J)
Js=M(50.0)
kw,km=10.0,25000.0
dvig=Dvig(Jr,kw,km)
mkl=1450.0
r1=0.25
r2=1.75
r4=0.5
d=0.15
crs=C(Jr,Js,2000.0,0.7)
#------------------------
Kl=ST_KL(mkl,r1,r2,d)
V=[Jr,Js,dvig]
E=[crs,dvig]
pi2=pi*2.0
t, dt, tkon = 0.0, 0.001, 10.50
bd=Bl('t','Mr','wr','Jr','x','dx','Mc','Mk','My')
while t<tkon:
   for i in V: i.f=0.0
   fi=Js.x[0]
   u=fmod(fi,pi2)
   Js.m=J+Kl.J(u)
   xx=Kl.f(u)
   pfk=Kl.df(u)
   Mk=0.5*(Kl.dJ(u))*Js.x[1]**2
   Mo=-mkl*r4*cos(u)
   Js.f=Mk+Mo
   for j in E: j.F()
   for i in V: i.step(dt)
   bd.add(t=t,Mr=Jr.f,wr=Jr.x[1],x=xx,dx=pfk,Mk=Mk)
   if t>=0.0:
      for i in V+E: i.add()
   t+=dt
#=======================================================
```

```
A,B=1000,460
tk = Tk()
tk.title("Одномассовый машинный агрегат")
f = Frame(tk)
f.pack()
c = Canvas(f, bg="White", width=A, height=B)
c.pack(expand=1, fill=BOTH)
#============================================================
clr=['blue','black','red','green','blue','gray','"#C06""#A64""#684""#AC3""#0DF','blue','green']
ww=(2,2,2,2,1,3,3,1,1,1,1,1,1,1)
cl=zip(clr,ww)
bg="White"
mg,xg,yg=0.004,400,200
desine_l(650,30, 300,300,(255,255,255),cl[0:],bd.L['Mr'],[bd.L['wr']],0,50,50,c,1,1)
LL=[bd.L['Mr']]
desine_l(20, 30, 600,300,(255,255,255),cl[1:],bd.L['t'], LL,0,50,50,c,1,1)
desine_l(20, 30, 600,300,(255,255,255),cl[3:],bd.L['t'], [bd.L['wr']],0,50,50,c,1,1)
ff2,tt='%s %15.5f %15.5f %s','
ff1='%15s %15.5f %s'
ff0='%15s %15.5f'
Lbd =[Bd(400,100,'w(t)','green',bg)]
Lbd+=[Bd(400,300,'M(t)','black',bg)]
Lbd+=[Bd(825,300,'w(M)','blue',bg)]
tt ='ПАРАМЕТРЫ\n'
tt+=ff1 % ('kw=',kw,'1/c\n')
tt+=ff1 % ('km=',km,'Нм')
Lbd+=[Bd(800,400,tt,'black',bg)]
tt ='РЕЗУЛЬТАТЫ\n'
tt+=ff2 % ('t =',min(bd.L['t']),max(bd.L['t']),'c \n')
tt+=ff2 % ('Wr=',min(bd.L['wr']),max(bd.L['wr']),'1/c \n')
tt+=ff2 % ('Mr=',min(bd.L['Mr']),max(bd.L['Mr']),'Нм \n')
tt+=ff2 % ('Mk=',min(bd.L['Mk']),max(bd.L['Mk']),'Нм \n')
Lbd+=[Bd(200,400,tt,'black',bg)]
for ibd in Lbd: ibd.show(c)
tk.mainloop()
```

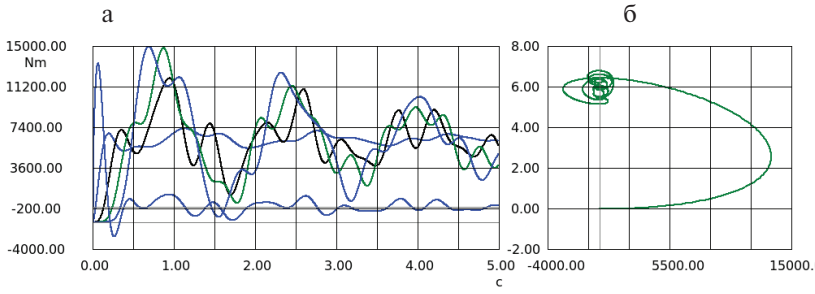


Рис. 19. Кинематическая цепь с упругой связью и двигателем:
а — момент $M(t)$ и скорость $w(t)$ ротора двигателя по времени;
б — фазовый портрет $w(M)$ двигателя

10. Библиотека программ

Библиотека программ включает элементы, используемые многократно другими программами. Это интерфейс, базы данных, численные методы и т. д.

Bsc_dinam.py

```
## bsc_dinam.py
from math import *
from bsc_grafik import *
global t
#============================================================
class D:
   Uo, Lpr, Rpr = 1.0, 0.01, 0.01
   def __init__(self):
      """Ke:=1; Tg:=0.05; Upr:=0;"""
      self.Upr,self.w,self.x,self.y=0.0,0.0,0.0,0.0
      self.Ke,self.Tg,self.bd=1.0,0.05,Bl('U')
   def step(self,dt):
      self.Upr=D.Uo-self.x*D.Rpr-self.y*D.Lpr
      self.y=((self.Upr-self.Ke*self.w)-self.x)/self.Tg
      self.x+=self.y*dt
   def add(self): self.bd.add(U=self.Upr)
class Dvig:
   def __init__(self,Jr,kw,km,b=1):
      self.d,self.Jr,self.kw,self.km=D(),Jr,kw,km
      self.f,self.b=0.0,b
```

```
      self.bd=Bl('f','x','v','a')
   def step(self,dt):
      self.d.step(dt)
   def F(self):
      self.d.w=self.Jr.x[1]/self.kw
      self.f=self.d.x*self.km
      self.Jr.f+=self.f
   def fzero(self):
      pass
   def add(self):
      if self.b:
         self.bd.add(f=self.f,x=self.Jr.x[0],v=self.Jr.x[1],a=self.Jr.x[2])
         self.d.add()
class Dvig_t(Dvig):
   def __init__(self,Jr,kw,km,tstart=0.0,tfinish=0.0,b=1):
      self.d,self.Jr,self.kw,self.km=D(),Jr,kw,km
      self.f,self.b=0.0,b
      self.bd=Bl('f','x','v','a')
      self.tstart=tstart
      self.tfinish=tfinish
   def step(self,dt):
      if t>=self.tstart and t<=self.tfinish:
         self.d.step(dt)
   def F(self):
      if (t>=self.tstart or self.tstart==0) and (t<=self.tfinish or self.tfinish==0):
         self.d.w=self.Jr.x[1]/self.kw
         self.f=self.d.x*self.km
         self.Jr.f+=self.f
      else:
         self.d.w=0.0
         self.f=0.0
         self.Jr.f=0.0
   def stop(self):
      self.Jr.f,self.Jr.x[1]=0.0,0.0
#=========================================================
class M:
   def __init__(self,m,b=1):
      self.m,self.f,self.x,=m,0.0,[0.0,0.0,0.0]
      self.bd=Bl('f','x','v','a')
      self.b=b
   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]
   def fzero(self):
      self.f=0.0
   def add(self):
      if self.b: self.bd.add(f=self.f,x=self.x[0],v=self.x[1],a=self.x[2])
#============================================================
class C:
   def __init__(self,a,b,c,r=0.0,dz=0.0,bb=1):
      self.a,self.b=a,b
      self.c,self.r=c,r
      self.dz=dz
      self.bb=bb
      self.f,self.ds,self.dv=0.0,0.0,0.0
      self.bd=Bl('f','ds','dv')
   def F(self):
      ds,dv=self.a.x[0]-self.b.x[0],self.a.x[1]-self.b.x[1]
      if  ds >  self.dz: ds-=self.dz
      elif ds < -self.dz: ds+=self.dz
      else: self.ds,self.dv=0.0,0.0
      self.f, self.ds, self.dv = ds*self.c+dv*self.r, ds, dv
      self.a.f-=self.f
      self.b.f+=self.f
   def add(self):
      if self.bb: self.bd.add(f=self.f,ds=self.ds,dv=self.dv)

def df(fx,fi,dfi):
   f2, f1 = fx(fi+dfi), fx(fi-dfi)
   return 0.5*(f2-f1)/dfi
def ddf(fx,fi,dfi):
   df2, df1 = df(fx,fi+dfi,dfi), df(fx,fi-dfi,dfi)
   return 0.5*(df2-df1)/dfi
class UM:
   def __init__(self,m,b=1):
      self.mx,self.my,self.mz=M(m),M(m),M(m)
      self.b=b
   def get_xv(self): return self.mx.x[0],self.mx.x[1],self.my.x[0],self.my.x[1],self.
mz.x[0],self.mz.x[1]
```

```
  def put_f(self,fx,fy,fz):
    self.mx.f+=fx
    self.my.f+=fy
    self.mz.f+=fz
  def fzero(self):
    for i in [self.mx,self.my,self.mz]: i.f=0.0
  def step(self,dt):
    for i in [self.mx,self.my,self.mz]: i.step(dt)
  def add(self):
    if self.b:
      for i in [self.mx,self.my,self.mz]: i.add()
class UC:
  def __init__(self,z1,z2,c=0.0,r=0.0):
    self.z1,self.z2=z1,z2
    x,y,z=self.z1.mx.x[0]-self.z2.mx.x[0],self.z1.my.x[0]-self.z2.my.x[0],self.z1.mz.x[0]-
self.z2.mz.x[0]
    self.l=pow(x**2+y**2+z**2,0.5)
    self.c,self.r=c,r
  def F(self):
    sx1,vx1,sy1,vy1,sz1,vz1=self.z1.get_xv()
    sx2,vx2,sy2,vy2,sz2,vz2=self.z2.get_xv()
    dsx,dvx=sx1-sx2,vx1-vx2
    dsy,dvy=sy1-sy2,vy1-vy2
    dsz,dvz=sz1-sz2,vz1-vz2
    l=pow(dsx**2+dsy**2+dsz**2,0.5)
    dl=l-self.l
    f=dl*self.c/l
    fx=f*dsx+dvx*self.r
    fy=f*dsy+dvy*self.r
    fz=f*dsz+dvz*self.r
    self.z1.put_f(-fx,-fy,-fz)
    self.z2.put_f( fx, fy, fz)
```

Bsc_grafik.py

```
## bsc_grafik.py
# -*- coding: utf-8 -*-
import sys
from Tkinter import *
from math   import *
import shelve
#-------------------------------------------------
def LayImagine(L=[(1,1),(10,10)], m=1.0, nx=100, ny=100,c=None,col="black",w=1):
   line=[]
   for (i,j) in L:
      ix,iy=nx+int(i/m),ny-int(j/m)
      line+=[(ix,iy)]
   c.create_line(line, fill=col, width=w)
class Bl:
   def __init__(self,*L):
      self.k,self.L,self.LL=L[0],{},{}
      for i in L:
         self.L[i]=[]
         self.LL[i]=[]
   def add(self,**L):            #L={}):
      for i,j in L.items():
         if i in self.L.keys(): self.L[i]+=[j]
   def adf(self,**L):
      for i,j in L.items(): self.L[i].insert(0,j)
   def addL(self,**L):
      for i,j in L.items():
         if i in self.L.keys(): self.LL[i]+=[j]
   def show(self,name='',*L):
      sf,s="""""",""
      for i in L:
         if i in self.L.keys(): s+="%18s;" % i
      sf+= s+"\n"
      for k in xrange(len(self.L[self.k])):
         s=""
         for i in L:
            if i in self.L.keys(): s+="%18.6f;" % self.L[i][k]
         sf+=s+"\n"
```

```
      f=open(name,'w')
      f.write(sf)
      f.close()
   def showL(self,m,x,y,c,col='black',w=1):
      for i in self.LL.values():
         for k in i: LayImagine(k,m,x,y,c,col,w)
   def showLk(self,k,m,x,y,c,col='black',w=1):
      for i in self.LL[k]: LayImagine(i,m,x,y,c,col,w)
#----------------------------------------
class Bd:
   def __init__(self,x,y,s,fg,bg):
      self.x,self.y,self.s,self.fg,self.bg=x,y,s,fg,bg
   def show(self,c):
      l=Label(c,text=self.s,fg=self.fg,bg=self.bg, justify='left', font=8)
      l.pack()
      c.create_window(self.x,self.y,window=l)
#----------------------------------------
def find_n(fi,x=[]):
   if fi<x[0] or x==[]: return 0
   l=len(x)
   for i in range(1,l):
      ii=i-1
      if fi<=x[i] and fi>x[ii]:
         return ii
   else: return l-1
def def_f(df,ly):
   ln=len(ly)-1
   ld=[]
   for i in range(1,ln):
      d=(ly[i+1]-ly[i-1])/df
      ld+=[d]
   lnn=ln-1
   ##d=(ly[1]-ly[-2])*0.5/df
   d=(ld[0]+ld[-1])*0.5
   ld.append(d)
   ld.insert(0,d)
   return ld
def def_v(ly):
   ln=len(ly)-1
   d=ly[1]-ly[ln]
   ld=[d]
   for i in range(1,ln):
```

```
      d=ly[i+1]-ly[i-1]
      ld+=[d]
   lnn=ln-1
   d=ly[0]-ly[lnn]
   ld.insert(0,d)
   ld.append(d)
   #ld+=[d]
   return ld
def take(name):
   f=open(name,'r')
   sf=f.read()
   f.close()
   ls=sf.split('\n')
   #----------------
   s0=ls[0]
   ls0=s0.split(';')
   l0=[]
   for i in ls0:
      j=i.strip()
      l0+=[j]
   #----------------
   d={}
   len0=len(l0)
   for i in range(len0): d[i]=[]
   for i in ls[1:]:
      lsi=i.split(';')
      k=0
      for j in lsi:
         jj=j.strip()
         try:
            ff=float(jj)
            d[k]+=[ff]
         except: print k,'--->',jj
         k+=1
   bl=Bl(l0[0])
   bl.L[l0[0]]=d[0]
   for i in range(len0)[1:]:
      bl.L[l0[i]]=d[i]
   return bl
#-----------------------------------------------------grafik
def maker(a=580, b=380,nx=50,ny=50,dx=10,dy=10):
   ixn,iyn,ixk,iyk=nx,ny,a+nx,b+ny
```

```
    ix,iy=ixn,iyn
    line=[ix,iy]
    while ix<ixk:
        line+=[ix,iyk,ix,iy]
        ix+=dx
        line+=[ix,iy]
    ix,iy=ixk,iyk
    line+=[ix,iy]
    while iy>iyn:
        line+=[ixn,iy,ix,iy]
        iy-=dy
        line+=[ix,iy]
    return line
def makel(a=580.0, b=380.0,x=[0.0, 580.0],y=[0.0,380.0],nx=5,ny=10):
    xmin, ymin, xmax, ymax = min(x), min(y), max(x), max(y)
    dx, dy = (xmax-xmin)/a, (ymax-ymin)/b
    if dx == 0.0 or dy == 0.0: return [nx,ny,x,y]
    line = []
    for i in xrange(len(x)):
        ix, iy = nx+int((x[i]-xmin)/dx), ny+int((ymax-y[i])/dy)
        line+=[ix,iy]
    return line
def makepl(a=400,b=400,fil=["black"],lt=[],lx=[],nx=20,ny=450,c=None):
    k,kl,mi=0,len(fil)-1,mashtab1(lt,lx,a,b,nx,ny)
    for i in mi:
        c.create_line(i,fill=fil[k])
        if k<kl: k+=1
        else: k=0
def mashtab1(mx,mmy,x,y,nx,ny):
    max_x,min_x=max(mx),min(mx)
    dx=(max_x-min_x)/x
    max_y,min_y=None,None
    for my in mmy:
        max_ay,min_ay=max(my),min(my)
        if max_y==None or max_y<max_ay: max_y=max_ay
        if min_y==None or min_y>min_ay: min_y=min_ay
    dy=(max_y-min_y)/y
    mmi=[]
    for my in mmy:
        mi=[]
        for i in range(len(mx)):
            ix,iy=int((mx[i]-min_x)/dx)+nx,y+ny-int((my[i]-min_y)/dy)
```

```
            kort=(ix,iy)
            mi.append(kort)
        mmi+=[mi]
    return mmi
def mashtab2(mx,mmy,x,y,by=0.0):
    max_x,min_x=max(mx),min(mx)
    dx=(max_x-min_x)/x
    if by==0:
        max_y,min_y=None,None
        for my in mmy:
            max_ay,min_ay=max(my),min(my)
            if max_y==None or max_y<max_ay: max_y=max_ay
            if min_y==None or min_y>min_ay: min_y=min_ay
        dy=(max_y-min_y)/y
    mmi=[]
    for my in mmy:
        mi=[]
        if by:
            max_y,min_y=max(my),min(my)
            dy=(max_y-min_y)/y*by
        for i in range(len(mx)):
            ix,iy=int((mx[i]-min_x)/dx),y-int((my[i]-min_y)/dy)
            if ix<1: ix=1
            if iy<1: iy=1
            if ix>x-1: ix=x-1
            if iy>y-1: iy=y-1
            kort=(ix,iy)
            mi.append(kort)
        mmi+=[mi]
    return mmi,max_x,min_x,max_y,min_y
def desine_f(n,a=4,b=3,rgb=(255,255,255),fil=[(0,0,0)],lt=[],lx=[],by=0,mes=
'proba',dix=5,diy=5):
    fill=(0,0,0)
    ab=[(1,1),(a-1,1),(a-1,b-1),(1,b-1),(1,1)]
    imp = Image.new('RGB',(a,b),rgb)
    draw = ImageDraw.Draw(imp, mode='RGB')
    x,y,ix,iy=a,b,0,0
    if x and y:
        mi,max_x,min_x,max_y,min_y=mashtab2(lt,lx,x,y,by)
        l=len(lx)
        dl=255/l
        (r,g,bl)=fill
```

```
      k=0
      kl=len(fil)-1
      for i in mi:
        draw.line(i,fil[k])
        if k<kl:
          k+=1
        else:
          k=0
      if mes:
        s='%8.3f;%8.3f' % (min_y,max_y)
        draw.text((10,5),mes+s, fill=(1,100,250))
      draw.line(ab,fill)
      while iy<y:
        iy+=diy
        draw.line([(1,iy),(x,iy)],fill)
      while ix<x:
        ix+=dix
        draw.line([(ix,1),(ix,y)],fill)
  imp.save(n,'gif')
def mashtab_l(mx,mmy,x,y,by=0.0,nx=1,ny=1):
  max_x,min_x=max(mx),min(mx)
  dx=(max_x-min_x)/x
  if by==0:
    max_y,min_y=None,None
    for my in mmy:
      max_ay,min_ay=max(my),min(my)
      if max_y==None or max_y<max_ay: max_y=max_ay
      if min_y==None or min_y>min_ay: min_y=min_ay
    dy=(max_y-min_y)/y
  mmi=[]
  for my in mmy:
    mi=[]
    if by:
      max_y,min_y=max(my),min(my)
      dy=(max_y-min_y)/y*by
    for i in range(len(mx)):
      ix,iy=int((mx[i]-min_x)/dx),y-int((my[i]-min_y)/dy)
      if ix<1: ix=1
      if iy<1: iy=1
      if ix>x-1: ix=x-1
      if iy>y-1: iy=y-1
      kort=(ix+nx,iy+ny)
```

```
            mi.append(kort)
        mmi+=[mi]
    return max_x,min_x,max_y,min_y,dy,dx,mmi
def desine_l(nx=1,ny=1,a=4,b=3,rgb=(255,255,255),fil=[(0,1)],lt=[],lx=[],
by=0,dix=5,diy=5,c=None,w=1,wr=1):
    if len(lt)<2: return
    x,y,ix,iy=a,b,0,0
    if x and y:
        max_x,min_x,max_y,min_y,dy,dx,mi=mashtab_l(lt,lx,x,y,by,nx,ny)
        if max_x==min_x or max_y==min_y or mi==[]: return
        l=len(lx)
        dl=255/l
        k,kl=0,len(fil)-1
        for i in mi:
            c.create_line(i,fill=fil[k][0],width=fil[k][1])
            if k<kl: k+=1
            else:   k=0
    rline=maker(a,b,nx,ny,dix,diy)
    c.create_line(rline, fill="black",width=w)
    if max_y*min_y<0 and by==0:
        c.create_line([nx,ny+b+int(min_y/dy),nx+a,ny+b+int(min_y/dy)], fill="gray",width=w)
    if max_x*min_x<0 and by==0:
        c.create_line([nx+a-int(max_x/dx),ny,nx+a-int(max_x/dx),ny+b], fill="gray",width=w)
    rline=maker(a,b,nx,ny,dix,diy)
    c.create_line(rline, fill="black",width=wr)
#-----------------------------------------------------
def mashtab_d(mm,x,y,nx=1,ny=1):
    lmax_y,lmin_y,lmax_x,lmin_x=[],[],[],[]
    for mx,my in mm:
        lmax_y+=[max(my)]
        lmin_y+=[min(my)]
        lmax_x+=[max(mx)]
        lmin_x+=[min(mx)]
    max_x,min_x,max_y,min_y=max(lmax_x),min(lmin_x),max(lmax_y),min(lmin_y)
    dx,dy=(max_x-min_x)/x,(max_y-min_y)/y
    mmi=[]
    for mx,my in mm:
        mi=[]
        for i in range(len(mx)):
            ix,iy=int((mx[i]-min_x)/dx),y-int((my[i]-min_y)/dy)
            px,py=ix+nx,iy+ny
            if px<1: px=1
```

```
            if py<1: py=1
            if px>=x+nx: px=x+nx-1
            if py>=y+ny: py=y+ny-1
            kort=(px,py)
            mi.append(kort)
        mmi+=[mi]
    return max_x,min_x,max_y,min_y,dy,dx,mmi
def desine_d(nx=1,ny=1,a=400,b=300,rgb=(255,255,255),fil=[(0,1),(0,1)],
          mm=[[(0.0,100.0),(0.0,1.0)],[(0.0,100.0),(0.0,1.0)]], dix=50,diy=50,c=None, w=1, wr=1):
    max_x,min_x,max_y,min_y,dy,dx,mmi=mashtab_d(mm,a,b,nx,ny)
    k,kl=0,len(fil)-1
    for mi in mmi:
        try:
            c.create_line(mi,fill=fil[k][0],width=fil[k][1])
            k+=1
            if k>kl: k=0
        except: pass
    if max_y*min_y<0: c.create_line([nx,ny+b+int(min_y/dy),nx+a,ny+b+int(min_y/
dy)], fill="gray",width=w)
    if max_x*min_x<0: c.create_line([nx+a-int(max_x/dx),ny,nx+a-int(max_x/
dx),ny+b], fill="gray",width=w)
    rline=maker(a,b,nx,ny,dix,diy)
    c.create_line(rline, fill="black", width=wr)
#----------------------------------------------------class
def F0(x,y,r):
    df=0.05*pi
    fk,f,L=2.0*pi+df,0.0,[]
    while f<fk:
        xc,yc=r*cos(f)+x,r*sin(f)+y
        L+=[(xc,yc)]
        f+=df
    return L
def F4(x,y,a,b,u=0.0):
    alf=atan(b/a)
    r=sqrt(a*a+b*b)
    dx1=r*cos(alf+u)
    dy1=r*sin(alf+u)
    dx2=r*cos(u-alf)
    dy2=r*sin(u-alf)
    return [(x+dx1,y+dy1),(x+dx2,y+dy2),(x-dx1,y-dy1),(x-dx2,y-dy2),(x+dx1,y+dy1)]
def F3(x,y,a,b):
    L=[(x,y),(x-a,y-b),(x+a,y-b),(x,y)]
```

```
    return L
def put_shelve(key,data,fname):
    d = shelve.open(fname)
    d[key] = data
    d.sync()
    d.close()
    return 1
def get_shelve(key,fname):
    d = shelve.open(fname)
    if d.has_key(key):
        data=d[key]
        d.close()
        return data
    else:
        d.close()
        return 0
def del_shelve(key,fname):
    d = shelve.open(fname)
    if d.has_key(key):
        del d[key]
        d.sync()
        d.close()
        return 1
    else:
        d.close()
        return 0

if __name__=="__main__":
    a={10:('name1', 'age1'),
       1 :('bsc', '52'),
       2: ('bsc-c', '12'),
    put_shelve('POCHO',a,'tm1')
    bb=get_shelve('POCHO','tm1')
          for i in bb.keys(): print i,bb[i]
```

Список библиографических ссылок

1. Теория механизмов и машин: учеб. пособие для студ. высш. учеб. заведений / М. З. Коловский, А. Н. Евграфов, Ю. А. Семенов, А. В. Слоущ. М. : Издательский центр «Академия», 2006. 560 с.
2. Белоконев И. М. Механика машин. Расчеты с применением ЭЦВМ. Киев: «Вища школа», 1978. 232 с.
3. Березин И. С., Жидков Н. П. Методы вычислений. Т. 1, 2. М. : Наука, 1966; М. : Физматгиз, 1962.
4. Лутц М. Изучаем Python. 3-е издание ; [пер. с англ.]. СПб. : Символ-Плюс, 2009. 848 с.
5. Дэвид М., Бизли Д. Язык программирования Python: справочник ; [пер. с англ.]. Киев : Изд-во «ДиаСофт», 2000. 336 с.

Оглавление

Предисловие 3
1. Плоские механизмы. Общие сведения 4
2. Кинематика плоских механизмов 8
    2.1. Группа Ассура первого вида 8
    2.2. Группа Ассура второго вида 13
    2.3. Группа Ассура третьего вида 17
3. Кулачковые механизмы 27
4. Зубчато-рычажные механизмы 34
5. Сборка нескольких групп Ассура 38
6. Примеры исследования механизмов 42
7. Влияние двигателя 54
8. Влияние приведенных к кривошипу сил и масс 57
9. Влияние упругих элементов 61
10. Библиотека программ 65
Список библиографических ссылок 78

Учебное издание

Буйначев Сергей Константинович
Баженов Евгений Евгеньевич
Троицкий Игорь Витальевич

МОДЕЛИРОВАНИЕ ДВИЖЕНИЯ И НАГРУЗОК ПЛОСКИХ МЕХАНИЗМОВ НА ЯЗЫКЕ PYTHON

Редактор О. В. Протасова
Верстка Е. В. Ровнушкиной

Подписано в печать 21.06.2019. Формат 60×84 1/16.
Бумага писчая. Цифровая печать. Усл. печ. л. 4,7.
Уч.-изд. л. 2,7. Тираж 40 экз. Заказ 208.

Издательство Уральского университета
Редакционно-издательский отдел ИПЦ УрФУ
620049, Екатеринбург, ул. С. Ковалевской, 5
Тел.: 8 (343) 375-48-25, 375-46-85, 374-19-41
E-mail: rio@urfu.ru

Отпечатано в Издательско-полиграфическом центре УрФУ
620083, Екатеринбург, ул. Тургенева, 4
Тел.: 8 (343) 358-93-06, 350-58-20, 350-90-13
Факс: 8 (343) 358-93-06
http://print.urfu.ru

9 785799 626853

БУЙНАЧЕВ СЕРГЕЙ КОНСТАНТИНОВИЧ
Кандидат технических наук, доцент кафедры металлургических и роторных машин ИНМТ УрФУ.
Преподаваемые дисциплины: «Математическое моделирование процессов и оборудования», «Прикладная механика», «Теория механизмов и машин».

БАЖЕНОВ ЕВГЕНИЙ ЕВГЕНЬЕВИЧ
Доктор технических наук, профессор. 11 лет возглавлял кафедру Автомобили и тракторы УГТУ–УПИ (ныне УрФУ). Директор института Автомобильного транспорта и технологических систем Уральского государственного лесотехнического университета (УГЛТУ).

ТРОИЦКИЙ ИГОРЬ ВИТАЛЬЕВИЧ
Кандидат технических наук, доцент кафедры металлургических и роторных машин ИНМТ УрФУ.
Преподаваемые дисциплины: «Проектирование полиграфических машин», «Механика», «Прикладная механика».