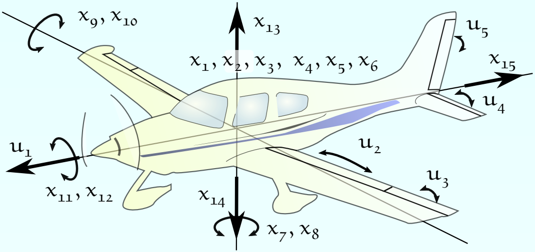
ПРИМЕНЕНИЕ КОМПЬЮТЕРНОЙ АЛГЕБРЫ В РЕАЛИЗАЦИИ АЛГОРИТМОВ УЛУЧШЕНИЯ

Черкашин Е.А., Бадмацыренова С.Б.

ИДСТУ СО РАН, ИРНИТУ, ИМЭИ ИГУ

«Винеровские чтения — 2015»
16 – 18 апреля 2015 г., Иркутск

Вектор-функция



Состояние, управление

$\vec{x} = \mathbf{x} = \langle x_1, x_2, \dots, x_n \rangle,$

$n = 15,$

$\vec{u} = \mathbf{u} = \langle u_1, u_2, \dots, u_m \rangle,$

$m = 5,$

$\vec{f} = \mathbf{f}.$

Уравнение движения (Задача Коши)

$$\dot{\mathbf{x}} = \frac{\mathbf{x}(t)}{dt} = \mathbf{f}(t, \mathbf{x}(t), \mathbf{u}(t)), \quad t \in T = [t_0, t_1],$$

$$\mathbf{x}(t_0) = \mathbf{x}_0, \quad \mathbf{x}(t) \in R^n, \ \mathbf{u}(t) \in R^m, \quad t \in T,$$

$$I(\mathbf{x}, \mathbf{u}) = \int_{t_0}^{t_1} f^0(t, \mathbf{x}(t), \mathbf{u}(t))dt \to \min .$$

Постановка задачи

Дискретный вариант уравнения движения (Задача Коши)

$$\mathbf{x}(t+1) = \mathbf{f}(t, \mathbf{x}(t), \mathbf{u}(t)), \quad t \in T = \{t_0, t_0+1, \ldots, t_1\}, \tag{1}$$

$$\mathbf{x}(t_0) = \mathbf{x}_0, \quad \mathbf{x}(t) \in R^n, \ \mathbf{u}(t) \in R^m, \quad t \in T, \tag{2}$$

$$I(\mathbf{x}, \mathbf{u}) = F(\mathbf{x}(t_1)) + \sum_{t_0}^{t_1-1} f^0(t, \mathbf{x}(t), \mathbf{u}(t)) \to \min. \tag{3}$$

Задача улучшения

Пусть заданы управление $\mathbf{u}^I(t)$ и соответствующее состояние $\mathbf{x}^I(t)$. Требуется найти $\mathbf{x}^{II}(t)$, $\mathbf{u}^{II}(t)$ такие, что

$$I(\mathbf{x}^{II}, \mathbf{u}^{II}) < I(\mathbf{x}^I, \mathbf{u}^I).$$

Алгоритм первого порядка

1. Задается начальное управление $\mathbf{u}^{\mathrm{I}}(t)$, из уравнения (1) и условий (2) определяется $\mathbf{x}^{\mathrm{I}}(t)$. Вычисляется $I(\mathbf{x}^{\mathrm{I}}, \mathbf{u}^{\mathrm{I}})$.
2. Из системы $(t) = \mathbf{H}_{\mathbf{x}},\ \boldsymbol{\psi}(t_1) = -\mathbf{F}_{\mathbf{x}}$, находим $\boldsymbol{\psi}(t)$,
   где $\boldsymbol{\psi}(t)$ — n-вектор,
   $\mathbf{H}_{\mathbf{u}}(t, \mathbf{x}, \boldsymbol{\psi}(t+1), \mathbf{u}) = \boldsymbol{\psi}^{T}(t+1) \times \mathbf{f}(t, \mathbf{x}, \mathbf{u}) - f^{0}(t, \mathbf{x}, \mathbf{u})$,
   производные функции H по $\mathbf{u}$ ($\mathbf{H}_{\mathbf{u}}$) находятся в точке $\left(t, \mathbf{x}^{\mathrm{I}}(t), \boldsymbol{\psi}\left(t+1\right), \mathbf{u}^{\mathrm{I}}(t)\right)$. Задается параметр α.
3. Из системы $\mathbf{x}(t+1) = \mathbf{f}(t, \mathbf{x}(t), \mathbf{u}^{\mathrm{II}}(t)), \mathbf{x}(t_0) = \mathbf{x}_0$, где $\mathbf{u}^{\mathrm{II}}(t) = \mathbf{u}^{\mathrm{I}}(t) + \alpha\mathbf{H}_{\mathbf{u}}$, вычисляется $\mathbf{x}^{\mathrm{II}}(t)$.
4. Новое управление и значение параметра α подсчитываются из решения задачи одномерной минимизации для функционала $I(\mathbf{x}^{\mathrm{II}}, \mathbf{u}^{\mathrm{II}}) \to \min\limits_{\alpha}$.
5. Если $I\left(\mathbf{x}^{\mathrm{II}}, \mathbf{u}^{\mathrm{II}}\right) \geqslant I\left(\mathbf{x}^{\mathrm{I}}, \mathbf{u}^{\mathrm{I}}\right)$ (улучшение не произошло), то уменьшаем α и переходим к следующей итерации, начиная с пункта 3.
6. Иначе, если $I\left(\mathbf{x}^{\mathrm{II}}, \mathbf{u}^{\mathrm{II}}\right) - I\left(\mathbf{x}^{\mathrm{I}}, \mathbf{u}^{\mathrm{I}}\right) > \varepsilon$, то переходим к следующей итерации, начиная с пункта 2. Значение ε — параметр точности.

Задание модели

```python
class LinModel1(Model):
    def __init__(self):
        X0=(1.0,)
        self.h = Ht
        self.num = int((1.0-0.0) / self.h)
        self.T = linspace(start=0.0, stop=1.0, num=self.num)
        self.t = arange(len(self.T))
        Model.__init__(self, X0=X0, U0=self.start_control())

    def start_control(self):
        U = [(0.0,) for t in self.t[:-1]]
        return array(U)

    def F(self, x):
        return 0.0

    def f(self, t, x, u):
        x0=x[0]; u0=u[0]
        return (x0+self.h*u0,)

    def f0(self, t, x, u):
        x0=x[0]; u0=u[0]
        return self.h * (x0*x0+u0*u0)
```

Реализация алгоритма улучшения

```
class Process(VFCalc):
    def __init__(self, model, alpha=1.0):
        VFCalc.__init__(self, model.N, model.M)
        t,x,u=self.v.t,self.v.x,self.v.u
        self.model=model; self.alpha=alpha
        self.v.f=model.f(t, x, u)
        self.v.f0=model.f0(t, x, u)
        self.v.F=model.F(x)

    def trajectory(self, U):
        x0=self.model.X0; X = [x0]
        for t, u in enumerate(U): # (0, u0), (1, u1)
            xn=self.model.f(t, X[t], u); X.append(xn)
        return array(X)

    def I(self, X, U):
        def _a(acc, t):
            return acc + self.model.f0(t, X[t], U[t])
        return reduce(_a, range(len(X)-1), self.model.F(X[-1]))

    def optimize(self, t, eps=0.001, iters=1000):
        Up=self.model.U0
        Xp=self.trajectory(Up)
        Ip=self.I(Xp,Up); it = 1
```

Реализация алгоритма улучшения (optimize)

```
while True:
    beta = self.beta
    Psi=self.Psi(t, Xp, Up, self.alpha)
    _H_u=self.H((self.v.u,), t[:-1], Xp[:-1], Up, Psi)
    while True:
        _dU=_H_u*alpha
        Un = Up + _dU
        Xn = self.trajectory(Un)
        In = self.I(Xn, Un)
        dI = Ip-In
        if abs(dI)<eps:
            return In, Xn, Un, it, "Opt"
        if iters<=0:
            return In, Xn, Un, it, "Nonoptimal"
        iters-=1; it+=1
        if In>=Ip:
            alpha/=2
            continue
        else:
            Xp, Up, Ip = Xn, Un, In
            break
```

Реализация алгоритма улучшения (psi, H)

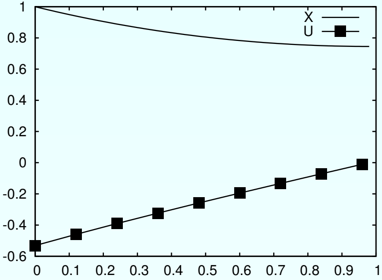
```
def Psi(self, t, X, U, alpha):
    v=self.v
    psie = -self.fun(v.F,(v.x,), t[-1], X[-1], U[-1])
    psi=[psie]; X=X[:-1]; t=t[:-1]
    _f0_x=self.fun(v.f0, (v.x,), t, X, U) # Vector context
    _f_x =self.fun(v.f, (v.x,), t, X, U)  # Vector context
    j=len(t)-1; p=psie
    while j>=1:
        i=t[j]; pp=p
        pn = dot(pp, _f_x[i]) - alpha*_f0_x[i]
        psi.append(pn); p=pn; j-=1
    psi=array(psi)
    return psi[::-1]

def H(self, vars, T, X, U, Psi, alpha = 1.0):
    f=self.fun(self.v.f, vars, T, X, U)
    f0=-self.fun(self.v.f0, vars, T, X, U)
    H = alpha * f0
    for psi,_H,_f,i in zip(Psi, H, f, range(len(H))):
        _H += dot(psi,_f); H[i]=_H
    return H
```

Реализация алгоритма улучшения (fun)

```
def fun(self, f, vars, T, X, U):
    code,df=self.code(f, *vars)
    X=numpy.atleast_1d(X)
    U=numpy.atleast_1d(U)
    if X.ndim>1:
        Xs=[X[:,i:i+1] for i in range(X.shape[1])]
        Us=[U[:,i:i+1] for i in range(U.shape[1])]
        m,args=True,(T,)+tuple(Xs+Us)
    else:
        m,args=False,(T,)+tuple(X)+tuple(U)
    rc=code(*args)
    rct=type(rc)
    rc=numpy.atleast_1d(rc)
    if type(T)==numpy.ndarray:
        try:
            if rct in [TupleType,ListType]:
                rc=rc.reshape(rc.shape[:-1])
                rc=rc.T
                return rc
        except ValueError: pass
        if T.shape[0]!=rc.shape[0]:
            nrc=numpy.zeros((len(T),)+rc.shape,dtype=float)
            nrc[:]=rc; rc=nrc
    return rc
```

Выполнение теста




```
I: 0.77666 iters: 47
```

```
def test_with_plot():
    m = LinModel1()
    p1=Process(m, beta=1.0)
    iters=2000
    eps=0.001
    print ("First-order process:")
    print ("-"*20)
    r1=I1, X1, U1, it1, _1 =
        p1.optimize(m.t, eps=eps,
            iters=iters)
    print (I1, "iters:", it1)
    gnuplot("test_with_plt", r1)
```

Дифференцирование вектор-функций

$$\mathbf{f} = \langle f^1, f^2, \ldots, f^n \rangle \ (1 \times n),$$
$$\mathbf{x} = \langle x_1, x_2, \ldots, x_n \rangle \ (1 \times n),$$
$$\mathbf{u} = \langle u_1, u_2, \ldots, u_m \rangle \ (1 \times m).$$

$\Longrightarrow$

$$\mathbf{f}_x = \begin{pmatrix} f^1_1 & f^2_1 & \cdots & f^n_1 \\ f^1_2 & f^2_2 & \cdots & f^n_2 \\ & & \ddots & \\ f^1_n & f^2_n & \cdots & f^n_n \end{pmatrix} \ (n \times n)$$

Производная $\mathbf{f}_u$ имеет размерность $(m \times n)$, $\mathbf{f}_{x,x}$ — $(n \times n \times n)$.

Пример вычисления производной вектор-функции

```
d=VFCalc(2,2)
x1,x2=Symbol('x1'),Symbol('x2')
u1,u2=Symbol('u1'),Symbol('u2')
y1=x1**2*u1+x2*u2**2
y2=x1**2*x2**2*u1**2*u2**2
res=(d.diff([y1,y2],
    [x1,x2], [u1,u2]))
pprint (res)
return
```

```
(((2*x1, 0), (0, 2*u2)),
 ((4*u1*u2**2*x1*x2**2,
   4*u1**2*u2*x1*x2**2),
  (4*u1*u2**2*x1**2*x2,
   4*u1**2*u2*x1**2*x2)))
```

Компьютерная алгебра: дифференцирование вектор-функций

```
class VFCalc(object):
    def __init__(self, N,M):
        self.N=N; self.M=M; self.v=Helper()
        self.v.x=[Symbol('x'+str(i+1)) for i in range(self.N)]
        self.v.u=[Symbol('u'+str(i+1)) for i in range(self.M)]
        self.v.t=Symbol('t')

    def diff1(self, f, var):
        if type(f) in [TupleType,ListType]:
            df=tuple([self.diff1(fi, var) for fi in f])
        else:
            df=tuple([diff(f, vi) for vi in var])
        if len(df)==1: df=df[0]
        return df

    def diff(self, f, *vars):
        cf=f
        for v in vars:
            cf=self.diff1(cf, v)
        return cf
```

Компьютерная алгебра: дополнительные функции

```
class VFCalc(object):
    # . . . . . . . .
    def subs(self, f, s):
        if type(f) not in [TupleType,ListType]:
            return f.subs(s) # common subs
        return tuple([self.subs(fi,s) for fi in f])#local subs

    def lambdify(self, f):
        l=[self.v.t]
        l.extend(self.v.x)
        l.extend(self.v.u)
        fl=lambdify(l, f, "numpy")
        return fl

    def code(self, f, *vars):
        df=self.diff(f, *vars)
        c=self.lambdify(df)
        return c,df
```

Заключение

Результаты

- Разработана первая версия библиотеки дифференцирования вектор-функций, поддерживающей библиотеки `numpy` и `scipy`;
- Использование библиотеки продемонстрировано на примере;
- Осуществляется реализация метода второго порядка.

Использование библиотеки позволяет повысить и точность и производительность вычислений.

Задачи на будущее

- На основе методов `diff`, `lambdify` и `fun` реализовать генераторы кода разных вариантов $\mathbf{f}$, $\mathbf{f}^{\circ}$, $\mathbf{F}$.
- Разработать эффективные процедуры для вычисления $\boldsymbol{\psi}$, $\mathbf{H_u}$ и $\mathbf{x}$.
- Разработать методику представления схемы алгоритма улучшения и странслировать ее реализацию в компилируемый язык (C, C++, Fortran и т. п.).

Спасибо за внимание!

Постоянный адрес презентации: https://github.com/eugeneai/paropt/raw/exp/talk-2015-04-17-wiener.pdf

Презентация сверстана в пакете LaTeX Beamer.