Министерство образования и науки РФ
Новосибирский Государственный Технический Университет
Кафедра ПМт

Лабораторная работа №3

«Вычисление ИМФ»

по курсу «Математические методы планирования эксперимента»

Факультет:	ПМИ
Группа:	ПММ-81
Студенты:	Михайлов А.А., Санина А.А.
Преподаватели:	Черникова О.С., Чубич В.М.

Новосибирск 2013

1. Условие задачи

Построить и исследовать процедуру вычисления ИМФ.

2. Ход работы:

2.1. Алгоритм вычисления информационной матрицы одноточечного плана

1) Для данного θ_i найти:

$$\begin{array}{ll}
\Phi,\ \dfrac{\partial \Phi}{\partial \theta_i},\ i=\overline{1,\, s} & Q,\ \dfrac{\partial Q}{\partial \theta_i},\ i=\overline{1,\, s} \\
\Psi,\ \dfrac{\partial \Psi}{\partial \theta_i},\ i=\overline{1,\, s} & R,\ \dfrac{\partial R}{\partial \theta_i},\ i=\overline{1,\, s} \\
\Gamma,\ \dfrac{\partial \Gamma}{\partial \theta_i},\ i=\overline{1,\, s} & \overline{x}\left(0\right),\ \dfrac{\partial \overline{x}\left(0\right)}{\partial \theta_i},\ i=\overline{1,\, s} \\
H,\ \dfrac{\partial H}{\partial \theta_i},\ i=\overline{1,\, s} & P\left(0\right),\ \dfrac{\partial P\left(0\right)}{\partial \theta_i},\ i=\overline{1,\, s}
\end{array}$$

Сформировать матрицу Ψ_A в соответствии с равенством:

$$\Psi_A = \begin{bmatrix} \Psi \\ \frac{\partial \Psi}{\partial \theta_1} \\ \dots \\ \frac{\partial \Psi}{\partial \theta_s} \end{bmatrix}. \tag{1}$$

2) Положить $M\left(\Theta\right)=0;\ P\left(0\,|\,0\right)=P\left(0\right);\ \frac{\partial P\left(0\,|\,0\right)}{\partial \theta_i}=\frac{\partial P\left(0\right)}{\partial \theta_i},\ i=\overline{1,\, s},\ t=0.$

3) Вычислить $\Sigma_A\left(t+1\,|\,t\right)$ по формуле:

$$\Sigma_A\left(t+1\,|\,t\right)=\begin{cases} 0, & t=0; \\ \Phi_A\left(t+1\,|\,t\right)\Sigma_A\left(t\,|\,t-1\right)\Phi_A^T\left(t+1\,|\,t\right)+K_A\left(t\right)B\left(t\right)K_A^T\left(t\right), & t>0. \end{cases} \tag{2}$$

4) Определить:

$$\overline{x}_A\left(t+1\,|\,t\right)=\begin{cases}
\begin{bmatrix} \Phi & 0 & \dots & 0 \\ \partial\Phi/\partial\theta_1 & \Phi & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \partial\Phi/\partial\theta_s & \dots & \dots & \Phi \end{bmatrix}\begin{bmatrix} \overline{x}\left(0\right) \\ \frac{\partial \overline{x}\left(0\right)}{\partial \theta_1} \\ \dots \\ \frac{\partial \overline{x}\left(0\right)}{\partial \theta_s} \end{bmatrix}+\begin{bmatrix} \Psi \\ \frac{\partial \Psi}{\partial \theta_1} \\ \dots \\ \frac{\partial \Psi}{\partial \theta_{s1}} \end{bmatrix}u\left(0\right), & t=0; \\
\underbrace{\begin{bmatrix} \Phi & 0 & \dots & 0 \\ \frac{\partial F}{\partial \theta_1}-\tilde{K}\left(t\right)\frac{\partial H}{\partial \theta_1} & \Phi-\tilde{K}\left(t\right)H & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \frac{\partial F}{\partial \theta_s}-\tilde{K}\left(t\right)\frac{\partial H}{\partial \theta_s} & \dots & \dots & \Phi-\tilde{K}\left(t\right)H \end{bmatrix}}_{F}\begin{bmatrix} \overline{x}\left(0\right) \\ \frac{\partial \overline{x}\left(0\right)}{\partial \theta_1} \\ \dots \\ \frac{\partial \overline{x}\left(0\right)}{\partial \theta_s} \end{bmatrix}+\begin{bmatrix} \Psi \\ \frac{\partial \Psi}{\partial \theta_1} \\ \dots \\ \frac{\partial \Psi}{\partial \theta_{s1}} \end{bmatrix}u\left(t\right)+\begin{bmatrix} \tilde{K}\left(t\right) \\ \frac{\partial \tilde{K}\left(t\right)}{\partial \theta_1} \\ \dots \\ \frac{\partial \tilde{K}\left(t\right)}{\partial \theta_s} \end{bmatrix}\varepsilon\left(t\right), & t>0.
\end{cases} \tag{3}$$

Где $\tilde{K}\left(t\right)=\Phi K\left(t\right)$.

5) Найти:

$$\begin{aligned}
P\left(t+1\,|\,t\right) &= \Phi P\left(t\,|\,t\right)\Phi^T+\Gamma Q\Gamma^T, \\
B\left(t+1\,|\,t\right) &= HP\left(t+1\,|\,t\right)H^T+R, \\
K\left(t+1\right) &= P\left(t+1\,|\,t\right)H^T B^{-1}\left(t+1\right), \\
P\left(t+1\,|\,t+1\right) &= \left[I-K\left(t+1\right)H\right]P\left(t+1\,|\,t\right), \\
\tilde{K}\left(t+1\right) &= \Phi K\left(t+1\right).
\end{aligned}$$

6) Сформировать матрицу $\Phi_A\left(t+2\,|\,t+1\right)$ в соответствии с F в (3).

7) Вычислить:

$$
\begin{aligned}
\frac{\partial P(t+1\,|\,t)}{\partial\theta_i} &= \frac{\partial\Phi}{\partial\theta_i}P(t|t)\,\Phi^T+\Phi\frac{\partial P(t\,|\,t)}{\partial\theta_i}\Phi^T+\Phi P(t\,|\,t)\,\frac{\partial\Phi^T}{\partial\theta_i}+\frac{\partial\Gamma}{\partial\theta_i}Q\Gamma^T+\Gamma\frac{\partial Q}{\partial\theta_i}\Gamma^T+\Gamma Q\frac{\partial\Gamma^T}{\partial\theta_i};\\
\frac{\partial B(t+1)}{\partial\theta_i} &= \frac{\partial H}{\partial\theta_i}P(t+1\,|\,t)\,H^T+H\frac{\partial P(t+1\,|\,t)}{\partial\theta_i}H^T+HP(t+1\,|\,t)\,\frac{\partial H^T}{\partial\theta_i}+\frac{\partial R}{\partial\theta_i};\\
\frac{\partial K(t+1)}{\partial\theta_i} &= \left[\frac{\partial P(t+1\,|\,t)}{\partial\theta_i}H^T+P(t+1\,|\,t)\,\frac{\partial H^T}{\partial\theta_i}-P(t+1\,|\,t)\,H^TB^{-1}(t+1)\,\frac{\partial B(t+1)}{\partial\theta_i}\right]B^{-1}(t+1);\\
\frac{\partial P(t+1\,|\,t+1)}{\partial\theta_i} &= [I-K(t+1)\,H]\,\frac{\partial P(t+1\,|\,t+1)}{\partial\theta_i}-\left[\frac{\partial K(t+1)}{\partial\theta_i}H+K(t+1)\,\frac{\partial H}{\partial\theta_i}P(t+1\,|\,t)\right];\\
\frac{\partial\tilde{K}(t+1)}{\partial\theta_i} &= \frac{\partial\Phi}{\partial\theta_i}K(t+1)+\Phi\frac{\partial K(t+1)}{\partial\theta_i}.
\end{aligned}
$$

8) Сформировать матрицу:

$$
K_A(t+1)=\begin{bmatrix}\tilde{K}(t)\\ \frac{\partial\tilde{K}(t)}{\partial\theta_1}\\ \dots\\ \frac{\partial\tilde{K}(t)}{\partial\theta_s}\end{bmatrix}. \tag{4}
$$

9) Вычислить $\Delta M(\Theta)$, используя формулу:

$$
\begin{aligned}
M_{ij}(\Theta) &= \sum_{t=0}^{N-1}\Bigg\{\mathrm{Sp}\left[C_0\left(\Sigma_A(t+1\,|\,t)+\overline{x}_A(t+1\,|\,t)\,\overline{x}_A^T(t+1\,|\,t)\right)C_0^T\frac{\partial H^T}{\partial\theta_i}B^{-1}(t+1)\,\frac{\partial H}{\partial\theta_i}\right]+\\
&+ \mathrm{Sp}\left[C_0\left(\Sigma_A(t+1\,|\,t)+\overline{x}_A(t+1\,|\,t)\,\overline{x}_A^T(t+1\,|\,t)\right)C_j^TH^TB^{-1}(t+1)\,H\right]+\\
&+ \mathrm{Sp}\left[C_i\left(\Sigma_A(t+1\,|\,t)+\overline{x}_A(t+1\,|\,t)\,\overline{x}_A^T(t+1\,|\,t)\right)C_0^T\frac{\partial H^T}{\partial\theta_j}B^{-1}(t+1)\,H\right]+\\
&+ \mathrm{Sp}\left[C_i\left(\Sigma_A(t+1\,|\,t)+\overline{x}_A(t+1\,|\,t)\,\overline{x}_A^T(t+1\,|\,t)\right)C_j^TH^TB^{-1}(t+1)\,H\right]+\\
&+ \frac{1}{2}\mathrm{Sp}\left[\frac{\partial B(t+1)}{\partial\theta_i}B^{-1}(t+1)\,\frac{\partial B(t+1)}{\partial\theta_j}B^{-1}(t+1)\right]\Bigg\},\ i,j=\overline{1,s}.
\end{aligned}
$$

10) Положить $M(\Theta)=M(\Theta)+\Delta M(\Theta)$.

11) Увеличить t на единицу. Если $t\le N-1$, перейти на шаг 3). В противном случае закончить процесс.

2.2. Текст программы на языке Python

```python
# -*- coding: utf-8 -*-

import numpy as np
from scipy import linalg as la


# Создать блочную матрицу - столбец [A, dA[0]..dA[N]]
def row_stack_it(A, dA):
    return np.row_stack([A, np.row_stack(dA)])


# Создать блочную матрицу - строку [A, dA[0]..dA[N]]
def col_stack_it(A, dA):
    return np.column_stack([A, np.column_stack(dA)])


# Создать блочную матрицу - строку [O..O, I, O..O], I на i+1 позиции. Размер блока n*n, число блоков s+1
def build_c(n, s, i):
    return np.column_stack([np.eye(n) if i == j else np.zeros((n, n)) for j in xrange(s + 1)])


class IMFSolver(object):
    def __init__(self, n, r, p, m, s, N):
        self.n = n                    # Размерность вектора состояний x
        self.r = r                    # Размерность вектора управления u
        self.p = p                    # Размерность вектора возмущений w
        self.m = m                    # Размерность вектора измерений y и вектора ошибки измерения v

        self.N = N                    # Число наблюдений: t=0..N
        self.s = s                    # Число параметров theta
        self._init_structures()

    def _init_structures(self):
        data = dict()
        data['n'] = self.n
        data['r'] = self.r
        data['p'] = self.p
        data['m'] = self.m

        data['N'] = self.N
        data['s'] = self.s
```

```

        # Матрица состояния
        data['Phi'] = np.ndarray((self.n, self.n))
        for i in xrange(self.s):
            data['diff(Phi,␣theta[%d])' % i] = np.ndarray((self.n, self.n))

        # Матрица управления
        data['Psi'] = np.ndarray((self.n, self.r))
        for i in xrange(self.s):
            data['diff(Psi,␣theta[%d])' % i] = np.ndarray((self.n, self.r))

        # Матрица возмущения
        data['Gamma'] = np.ndarray((self.n, self.p))
        for i in xrange(self.s):
            data['diff(Gamma,␣theta[%d])' % i] = np.ndarray((self.n, self.p))

        # Матрица измерений
        data['H'] = np.ndarray((self.m, self.n))
        for i in xrange(self.s):
            data['diff(H,␣theta[%d])' % i] = np.ndarray((self.m, self.n))

        data['Q'] = np.ndarray((self.p, self.p))
        for i in xrange(self.s):
            data['diff(Q,␣theta[%d])' % i] = np.ndarray((self.p, self.p))

        data['R'] = np.ndarray((self.m, self.m))
        for i in xrange(self.s):
            data['diff(R,␣theta[%d])' % i] = np.ndarray((self.m, self.m))

        data['x(0)'] = np.ndarray((self.n, 1))
        for i in xrange(self.s):
            data['diff(x(0),␣theta[%d])' % i] = np.ndarray((self.n, 1))

        data['P(0)'] = np.ndarray((self.n, self.n))
        for i in xrange(self.s):
            data['diff(P(0),␣theta[%d])' % i] = np.ndarray((self.n, self.n))

        data['M'] = np.ndarray((self.s, self.s))

        for i in xrange(self.N):
            data['u(%d)' % i] = np.ndarray((self.r, 1))

        self.data = data

    def set_Phi(self, A):
        self.data['Phi'][:, :] = A

    def set_diff_Phi_theta(self, A, i):
        self.data['diff(Phi,␣theta[%d])' % i][:, :] = A

    def set_Psi(self, A):
        self.data['Psi'][:, :] = A

    def set_diff_Psi_theta(self, A, i):
        self.data['diff(Psi,␣theta[%d])' % i][:, :] = A

    def set_Gamma(self, A):
        self.data['Gamma'][:, :] = A

    def set_diff_Gamma_theta(self, A, i):
        self.data['diff(Gamma,␣theta[%d])' % i][:, :] = A

    def set_H(self, A):
        self.data['H'][:, :] = A

    def set_diff_H_theta(self, A, i):
        self.data['diff(H,␣theta[%d])' % i][:, :] = A

    def set_Q(self, A):
        self.data['Q'][:, :] = A

    def set_diff_Q_theta(self, A, i):
        self.data['diff(Q,␣theta[%d])' % i][:, :] = A

    def set_R(self, A):
        self.data['R'][:, :] = A

    def set_diff_R_theta(self, A, i):
        self.data['diff(R,␣theta[%d])' % i][:, :] = A

    def set_x0(self, A):
        self.data['x(0)'][:, :] = A

    def set_diff_x0_theta(self, A, i):
        self.data['diff(x(0),␣theta[%d])' % i][:, :] = A

    def set_P0(self, A):
        self.data['P(0)'][:, :] = A

    def set_diff_P0_theta(self, A, i):
        self.data['diff(P(0),␣theta[%d])' % i][:, :] = A

    def set_u(self, A, t):
        self.data['u(%d)' % t][:, :] = A
```

```
    def solve(self):
        d = self.data
        # Шаг 1. Сформировать матрицу Psi_A в соответствии с равенством (2.77)
        d['Psi_A'] = row_stack_it(d['Psi'], [d['diff(Psi, theta[%d])' % i] for i in xrange(self.s)])

        # Шаг 2.
        d['M(Theta)'] = np.ndarray((self.s, self.s))
        d['M(Theta)'][:, :] = 0.0
        d['P(0|0)'] = d['P(0)']
        for i in xrange(self.s):
            d['diff(P(0|0), theta[%d])' % i] = d['diff(P(0), theta[%d])' % i]
        t = 0

        while t < self.N:
            self.step3(d, t)
            self.step4(d, t)
            self.step5(d, t)
            self.step6(d, t)
            self.step7(d, t)
            self.step8(d, t)
            self.step9(d, t)
            # Шаг 10. Положить M(Theta) = M(Theta) + delta M(Theta)
            d['M(Theta)'] += d['delta_M(Theta)']

            # Шаг 11. Увеличить t на единицу. Если t <= N-1, перейти на шаг 3. В противном случае закончить процесс
            t += 1
        return d['M(Theta)']

    def step3(self, d, t):
        # Шаг 3. Вычислить Sigma_A(t + 1|t) по формуле (2.82), если t = 0, иначе по формуле (2.84)
        if t == 0:
            d['Sigma_A(%d|%d)' % (t + 1, t)] = 0
        else:
            Phi = d['Phi_A(%d|%d)' % (t + 1, t)]
            Sigma = d['Sigma_A(%d|%d)' % (t, t - 1)]
            K = d['K_A(%d)' % t]
            B = d['B(%d)' % t]
            d['Sigma_A(%d|%d)' % (t + 1, t)] = np.dot(np.dot(Phi, Sigma), Phi.transpose()) + np.dot(np.dot(K, B), K.transpose())

    def step4(self, d, t):
        # Шаг 4. Определить x(t + 1|t) при помощи выражения (2.81)
        if t == 0:
            P1 = row_stack_it(d['Phi'], [d['diff(Phi, theta[%d])' % i] for i in xrange(self.s)])
            x0 = d['x(0)']
            z0 = np.ndarray((self.n, 1))
            z0[:, :] = 0
            P2 = row_stack_it(z0, [np.dot(d['Phi'], d['diff(x(0), theta[%d])' % i]) for i in xrange(self.s)])
            d['x_A(%d|%d)' % (t + 1, t)] = np.dot(P1, x0) + P2 + np.dot(d['Psi_A'], d['u(0)'])
        else:
            d['x_A(%d|%d)' % (t + 1, t)] = \
                np.dot(d['Phi_A(%d|%d)' % (t + 1, t)], d['x_A(%d|%d)' % (t, t - 1)]) + \
                np.dot(d['Psi_A'], d['u(%d)' % t])

    def step5(self, d, t):
        # Шаг 5.Найти P(t + 1|t), B(t + 1), K(t + 1), P(t + 1|t + 1), Kt(t + 1),
        # используя соотношения (2.9), (2.11), (2.12), (2.14) и (2.73)
        self.step5_P10(d, t)
        self.step5_B(d, t)
        self.step5_K(d, t)
        self.step5_P11(d, t)
        self.step5_Kt(d, t)

    def step5_P10(self, d, t):
        Phi = d['Phi']
        P = d['P(%d|%d)' % (t, t)]
        Gamma = d['Gamma']
        Q = d['Q']
        d['P(%d|%d)' % (t + 1, t)] = np.dot(np.dot(Phi, P), Phi.transpose()) + np.dot(np.dot(Gamma, Q), Gamma.transpose())

    def step5_B(self, d, t):
        H = d['H']
        P = d['P(%d|%d)' % (t + 1, t)]
        R = d['R']
        d['B(%d)' % (t + 1)] = np.dot(np.dot(H, P), H.transpose()) + R

    def step5_K(self, d, t):
        P = d['P(%d|%d)' % (t + 1, t)]
        H = d['H']
        B = d['B(%d)' % (t + 1)]
        d['K(%d)' % (t + 1)] = np.dot(np.dot(P, H.transpose()), la.inv(B))

    def step5_P11(self, d, t):
        I = np.eye(self.n)
        K = d['K(%d)' % (t + 1)]
        H = d['H']
        P = d['P(%d|%d)' % (t + 1, t)]
        d['P(%d|%d)' % (t + 1, t + 1)] = np.dot((I - np.dot(K, H)), P)

    def step5_Kt(self, d, t):
```

```
        Phi = d['Phi']
        K = d['K(%d)' % (t + 1)]
        d['Kt(%d)' % (t + 1)] = np.dot(Phi, K)

    def step6(self, d, t):
        # Шаг 6. Сформировать матрицу Phi_A(t+2|t+1) в соответствии с (2.76)
        First_col = row_stack_it(d['Phi'], [d['diff(Phi, theta[%d])' % i] -
                                            np.dot(d['Kt(%d)' % (t + 1)], d['diff(H, theta[%d])' % i]) for i in xrange(self.s)])
        Phi = d['Phi']
        Kt = d['Kt(%d)' % (t + 1)]
        H = d['H']
        Remain_cols = [row_stack_it(np.zeros((self.n, self.n)), [Phi - np.dot(Kt, H) if i == j else np.zeros((self.n, self.n))
                                                                  for j in xrange(self.s)]) for i in xrange(self.s)]
        d['Phi_A(%d|%d)' % (t + 2, t + 1)] = col_stack_it(First_col, Remain_cols)

    def step7(self, d, t):
        # Шаг 7. Вычислить производные для P(t+1|t), B(t+1), K(t+1), P(t+1|t+1), Kt(t+1) по формулам (2.85) - (2.89)
        self.step7_P10(d, t)
        self.step7_B(d, t)
        self.step7_K(d, t)
        self.step7_P11(d, t)
        self.step7_Kt(d, t)

    def step7_P10(self, d, t):
        for i in xrange(self.s):
            dPhi = d['diff(Phi, theta[%d])' % i]
            P = d['P(%d|%d)' % (t, t)]
            Phi = d['Phi']
            dP = d['diff(P(%d|%d), theta[%d])' % (t, t, i)]
            dGamma = d['diff(Gamma, theta[%d])' % i]
            Q = d['Q']
            Gamma = d['Gamma']
            dQ = d['diff(Q, theta[%d])' % i]
            d['diff(P(%d|%d), theta[%d])' % (t + 1, t, i)] = \
                np.dot(np.dot(dPhi, P), Phi.transpose()) + \
                np.dot(np.dot(Phi, dP), Phi.transpose()) + \
                np.dot(np.dot(Phi, P), dPhi.transpose()) + \
                np.dot(np.dot(dGamma, Q), Gamma.transpose()) + \
                np.dot(np.dot(Gamma, dQ), Gamma.transpose()) + \
                np.dot(np.dot(Gamma, Q), dGamma.transpose())

    def step7_B(self, d, t):
        for i in xrange(self.s):
            dH = d['diff(H, theta[%d])' % i]
            P = d['P(%d|%d)' % (t + 1, t)]
            H = d['H']
            dP = d['diff(P(%d|%d), theta[%d])' % (t + 1, t, i)]
            dR = d['diff(R, theta[%d])' % i]
            d['diff(B(%d), theta[%d])' % (t + 1, i)] = \
                np.dot(np.dot(dH, P), H.transpose()) + \
                np.dot(np.dot(H, dP), H.transpose()) + \
                np.dot(np.dot(H, P), dH.transpose()) + dR

    def step7_K(self, d, t):
        for i in xrange(self.s):
            dP = d['diff(P(%d|%d), theta[%d])' % (t + 1, t, i)]
            H = d['H']
            P = d['P(%d|%d)' % (t + 1, t)]
            dH = d['diff(H, theta[%d])' % i]
            B = d['B(%d)' % (t + 1)]
            dB = d['diff(B(%d), theta[%d])' % (t + 1, i)]
            d['diff(K(%d), theta[%d])' % (t + 1, i)] = np.dot(
                np.dot(dP, H.transpose()) +
                np.dot(P, dH.transpose()) -
                np.dot(np.dot(np.dot(P, H.transpose()), la.inv(B)), dB), la.inv(B))

    def step7_P11(self, d, t):
        for i in xrange(self.s):
            I = np.eye(self.n)
            K = d['K(%d)' % (t + 1)]
            H = d['H']
            dP = d['diff(P(%d|%d), theta[%d])' % (t + 1, t, i)]
            dK = d['diff(K(%d), theta[%d])' % (t + 1, i)]
            dH = d['diff(H, theta[%d])' % i]
            P = d['P(%d|%d)' % (t + 1, t)]
            d['diff(P(%d|%d), theta[%d])' % (t + 1, t + 1, i)] = \
                np.dot((I - np.dot(K, H)), dP) - \
                np.dot(np.dot(dK, H) + np.dot(K, dH), P)

    def step7_Kt(self, d, t):
        for i in xrange(self.s):
            dPhi = d['diff(Phi, theta[%d])' % i]
            K = d['K(%d)' % (t + 1)]
            Phi = d['Phi']
            dK = d['diff(K(%d), theta[%d])' % (t + 1, i)]
            d['diff(Kt(%d), theta[%d])' % (t + 1, i)] = np.dot(dPhi, K) + np.dot(Phi, dK)

    def step8(self, d, t):
        # Шаг 8. Сформировать матрицу K_A(t+1) в соответствии с (2.78)
        d['K_A(%d)' % (t + 1)] = row_stack_it(d['Kt(%d)' % (t + 1)],
```

```
                                                          [d['diff(Kt(%d), theta[%d])' % (t + 1, i)] for i in xrange(self.s)])

    def step9(self, d, t):
        # Шаг 9. Используя выражение (2.72), получить приращение deltaM(Theta), отвечающее текущему значению t.
        deltaM = np.ndarray((self.s, self.s))
        C0 = build_c(self.n, self.s, 0)
        Sigma_A = d['Sigma_A(%d|%d)' % (t + 1, t)]
        x_A = d['x_A(%d|%d)' % (t + 1, t)]
        B = d['B(%d)' % (t + 1)]
        H = d['H']
        for i in xrange(self.s):
            Ci = build_c(self.n, self.s, i + 1)
            dHi = d['diff(H, theta[%d])' % i]
            dBi = d['diff(B(%d), theta[%d])' % (t + 1, i)]
            for j in xrange(self.s):
                Cj = build_c(self.n, self.s, j + 1)
                dHj = d['diff(H, theta[%d])' % j]
                dBj = d['diff(B(%d), theta[%d])' % (t + 1, j)]

                deltaM[i, j] = \
                    np.trace(np.dot(np.dot(np.dot(np.dot(np.dot(C0, (Sigma_A + np.dot(x_A, x_A.transpose()))), C0.transpose()), dHj.transpose()), la.inv(B)), dHi)) + \
                    np.trace(np.dot(np.dot(np.dot(np.dot(np.dot(C0, (Sigma_A + np.dot(x_A, x_A.transpose()))), Cj.transpose()), H.transpose()), la.inv(B)), dHi)) + \
                    np.trace(np.dot(np.dot(np.dot(np.dot(np.dot(Ci, (Sigma_A + np.dot(x_A, x_A.transpose()))), C0.transpose()), dHj.transpose()), la.inv(B)), H)) + \
                    np.trace(np.dot(np.dot(np.dot(np.dot(np.dot(Ci, (Sigma_A + np.dot(x_A, x_A.transpose()))), Cj.transpose()), H.transpose()), la.inv(B)), H)) + \
                    0.5 * np.trace(np.dot(np.dot(dBi, la.inv(B)), np.dot(dBj, la.inv(B))))
        d['delta_M(Theta)'] = deltaM


def main():
    N = 20

    solver = IMFSolver(n=2, r=1, p=2, m=1, s=2, N=N)

    theta = [0.56, 0.48]

    solver.set_Phi([[1.0, 1.0], [-0.5, 0.0]])
    solver.set_diff_Phi_theta([[0.0, 0.0], [0.0, 0.0]], 0)
    solver.set_diff_Phi_theta([[0.0, 0.0], [0.0, 0.0]], 1)

    solver.set_Psi([[theta[0]], [theta[1]]])
    solver.set_diff_Psi_theta([[1.0], [0.0]], 0)
    solver.set_diff_Psi_theta([[0.0], [1.0]], 1)

    solver.set_Gamma([[1.0, 0.0], [0.0, 1.0]])
    solver.set_diff_Gamma_theta([[0.0, 0.0], [0.0, 0.0]], 0)
    solver.set_diff_Gamma_theta([[0.0, 0.0], [0.0, 0.0]], 1)

    solver.set_H([[1.0, 0.0]])
    solver.set_diff_H_theta([[0.0, 0.0]], 0)
    solver.set_diff_H_theta([[0.0, 0.0]], 1)

    solver.set_Q([[0.07, 0.0], [0.0, 0.07]])
    solver.set_diff_Q_theta([[0.0, 0.0], [0.0, 0.0]], 0)
    solver.set_diff_Q_theta([[0.0, 0.0], [0.0, 0.0]], 1)

    solver.set_R([[0.02]])
    solver.set_diff_R_theta([[0.0]], 0)
    solver.set_diff_R_theta([[0.0]], 1)

    solver.set_x0([[0.0], [0.0]])
    solver.set_diff_x0_theta([[0.0], [0.0]], 0)
    solver.set_diff_x0_theta([[0.0], [0.0]], 1)

    solver.set_P0([[0.1, 0.0], [0.0, 0.1]])
    solver.set_diff_P0_theta([[0.0, 0.0], [0.0, 0.0]], 0)
    solver.set_diff_P0_theta([[0.0, 0.0], [0.0, 0.0]], 1)

    for i in xrange(N):
        solver.set_u([[1.0]], i)

    M = solver.solve()
    print M
    print la.det(M)
    print -np.log(la.det(M))


if __name__ == '__main__':
    main()
```

2.3. Результат работы программы

- $\begin{pmatrix} x_1(t+1) \\ x_2(t+1) \end{pmatrix} = \Phi \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} + \Psi u(t) + \begin{pmatrix} w_1(t) \\ w_2(t) \end{pmatrix}$,
  $y(t+1) = x_1(t+1) + v(t+1)$;
- $Q = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, $R = 0.02$,
- $u(0) = u(1) = \ldots = u(N-1) = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$;
- $P(0) = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix}$, $\overline{x}(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$;
- $N = 20$;
- $\theta^* = \begin{pmatrix} \theta_1^* \\ \theta_2^* \end{pmatrix} = \begin{pmatrix} 0.56 \\ 0.48 \end{pmatrix}$.

В ходе выполнения программы, положим:

$$\Phi = \begin{pmatrix} 1 & 1 \\ -0.5 & 0 \end{pmatrix}, \qquad Q = 1,$$

$$\Psi = \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix}, \qquad R = 0.02,$$

$$\Gamma = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \qquad \overline{x}(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

$$H = \begin{pmatrix} 1 & 0 \end{pmatrix}, \qquad P(0) = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.1 \end{pmatrix}.$$

В результате было получено:

- $\det(M) = 7.77$.

3. Вывод

В ходе лабораторной работы была проведена разработка процедуры вычисления ИМФ для дискретного плана.