УДК 004

МОДЕЛИРОВАНИЕ И АНАЛИЗ СИСТЕМ СЛУЧАЙНОЙ СТРУКТУРЫ С ИСПОЛЬЗОВАНИЕМ ФИЛЬТРА ЧАСТИЦ

А. А. Мельников[1], аспирант гр. АЭР242, I курс
Научный руководитель: Павлов В.И.[1], д.т.н., профессор
ФГБОУ ВО «Тамбовский государственный технический университет», Тамбов

Аннотация.

Целью данной работы являлось разработать и анализировать метод оценки состояния нелинейных динамических систем с изменяющимися параметрами на основе фильтра частиц. Программная реализация метода и численный эксперимент, который демонстрирует эффективность предложенного подхода, выполнена на языке программирования Python. В результате исследования были проведены математическое обоснование метода, алгоритмическая реализация и численный эксперимент.

Ключевые слова: гаусовский шум, случайная выборка, системы случайной структуры, фильтр частиц.

Введение

Причина, по которой информационно-измерительные системы обладают случайной структурой кроется в наличии неопределённостей в параметрах этих систем и внешних воздействиях на ИИС. Такие условия вызывают проблемы оценки состояния системы на основе неполных и зашумленных данных. Классические методы фильтрации, такие как фильтр Калмана, эффективно работают только в случае линейных систем с гауссовскими шумами. Однако, в случаях нелинейных зависимостей с негауссовскими распределениями такие методы становятся менее сложными и применимыми [2]. Действенным подходом к фильтрации в таких условиях является применение фильтра частиц. Подход заключается в вероятностном моделировании и аппроксимации распределения состояний с помощью увеличения случайных выборок (частиц) [3].

Данное исследование направлено на разработку и анализ метода оценки состояния нелинейных динамических систем с изменяющимися параметрами на основе фильтра частиц.

Цель применения фильтра частиц, заключается в возможности оценивать состояние системы даже в сложных формах шума и нелинейных процессах.

Материалы и методы

1. Математическая модель

Уравнение состояния динамики системы имеет вид:

$$x_k = f(x_{k-1}, \theta_k, w_k)$$

где

x_k – состояние системы в текущий момент времени,

θ_k – изменяющиеся параметры системы,

w_k – ковариация шума процесса.

Уравнение наблюдений за состоянием системы:

$$y_k = h(x_k, v_k)$$

где

y_k – наблюдение за состоянием системы,

v_k – ковариация шума измерений.

Цель фильтра – на основе ряда наблюдений $y_{1:k}$ оценить экспериментально полученное распределение $p\{x_k|y_{1:k}\}$ [4].

2. Алгоритм фильтра частиц

Работа фильтра частиц основана на байесовской фильтрации методом Монте-Карло.

Алгоритм работает в пять этапов: инициализация, предсказание, обновление весов, обновление весов, ресемплинг, оценка состояния.

- **Инициализация:** согласно начальному распределению создаётся N количество частиц $\left\{x_k^{(i)}\right\}$.
- **Предсказание:** с учётом динамики системы, происходит обновление частиц:

$$x_k^{(i)} = f\left(x_{k-1}^{(i)}, \theta_k, w_k^{(i)}\right).$$

- **Обновление весов:** вычисляется вес $w_k^{(i)}$ для каждой частицы

$$w_k^{(i)} \propto p\left(y_k \middle| x_k^{(i)}\right).$$

- **Ресемплинг:** выбираются новые частицы, вероятности которых, пропорциональны весам этих частиц [1].
- **Оценка состояния:** выполняется вычисление средневзвешенного значение:

$$\hat{x}_k = \sum_{i=1}^{N} w_k^{(i)} x_k^{(i)})$$

3. Численный эксперимент

Задача:

С помощью фильтра частиц, выполнить оценку скрытого состояния системы на каждом шаге. Сравнить с истинными значениями и наблюдениями.

Ход решения:

Рассматривается нелинейная динамическая система, заданная уравнением:

$$x_k = 0{,}5x_{k-1} + \frac{25x_{k-1}}{1 + x_{k-1}^2} + 8\cos(1{,}2\theta_k) + w_k$$

где

- x_k – состояние системы в текущий момент времени;
- θ_k – изменяющийся внешний параметр, увеличивающийся на 0,1 на каждом шаге;
- $w_k \sim N(0,1,0)$ – шум процесса (нормальный закон).

$$y_k = \frac{x_k^2}{20} + v_k,$$

где

- $v_k \sim N(0,1,0)$ – шум измерений.

4. Программная реализация

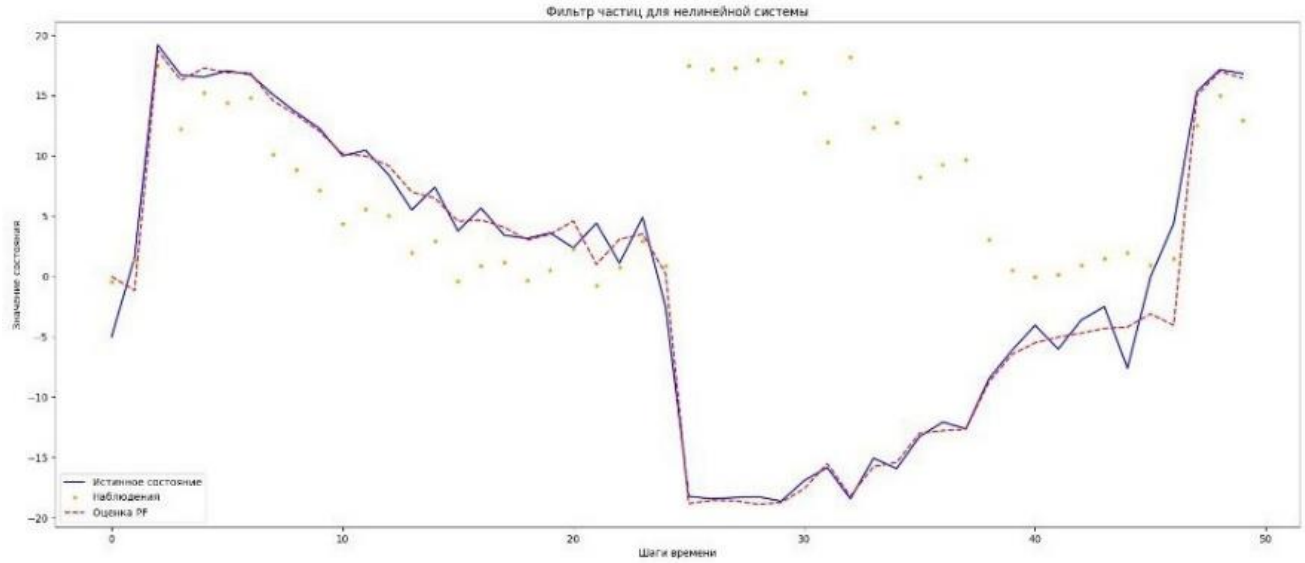
```
import numpy as np
import matplotlib.pyplot as plt

# Динамическая система
def system_dynamics(x, theta, noise):
    return 0.5 * x + (25 * x) / (1 + x**2) + 8 * np.cos(1.2 * theta) + noise

# Функция наблюдения
def measurement_function(x, noise):
    return (x**2) / 20 + noise

# Фильтр частиц
class ParticleFilter:
    def __init__(self, num_particles, process_noise, measurement_noise):
        self.num_particles = num_particles
        self.particles = np.random.uniform(-10, 10, num_particles)
        self.weights = np.ones(num_particles) / num_particles
        self.process_noise = process_noise
        self.measurement_noise = measurement_noise

    def predict(self, theta):
        self.particles = system_dynamics(self.particles, theta, np.random.nor-
mal(0, self.process_noise, self.num_particles))

    def update(self, measurement):
        predicted_measurements = measurement_function(self.particles, np.ze-
ros(self.num_particles))
        self.weights = np.exp(-0.5 * ((measurement - predicted_measurements) /
self.measurement_noise) ** 2)
        self.weights += 1e-300  # защита от деления на ноль
        self.weights /= np.sum(self.weights)

    def resample(self):
        indices = np.random.choice(range(self.num_particles), size=self.num_par-
ticles, p=self.weights)
        self.particles = self.particles[indices]
        self.weights.fill(1.0 / self.num_particles)

    def estimate(self):
        return np.sum(self.particles * self.weights)
```

```
# Параметры моделирования
num_steps = 50
num_particles = 1000
process_noise = 1.0
measurement_noise = 1.0

theta_values = np.arange(0, num_steps * 0.1, 0.1)
true_states = np.zeros(num_steps)
measurements = np.zeros(num_steps)
filtered_estimates = np.zeros(num_steps)

# Инициализация
true_states[0] = -5
measurements[0] = measurement_function(true_states[0], np.random.normal(0, meas-
urement_noise))
pf = ParticleFilter(num_particles, process_noise, measurement_noise)

# Моделирование и фильтрация
for k in range(1, num_steps):
    true_states[k] = system_dynamics(true_states[k-1], theta_values[k], np.ran-
dom.normal(0, process_noise))
    measurements[k] = measurement_function(true_states[k], np.random.normal(0,
measurement_noise))

    pf.predict(theta_values[k])
    pf.update(measurements[k])
    pf.resample()
    filtered_estimates[k] = pf.estimate()

# Визуализация
plt.figure(figsize=(10, 5))
plt.plot(true_states, label="Истинное состояние", color='blue')
plt.plot(measurements, "o", label="Наблюдения", color='orange', markersize=3)
plt.plot(filtered_estimates, label="Оценка PF", color='red', linestyle='dashed')
plt.legend()
plt.xlabel("Шаги времени")
plt.ylabel("Значение состояния")
plt.title("Фильтр частиц для нелинейной системы")
plt.show()
```

Результаты и обсуждения

В результате исследования было проведено математическое обоснование метода, алгоритмическая реализация и численный эксперимент. Программная реализация метода продемонстрировала эффективность предложенного подхода.

Результат выполнения программы:



Заключение

Применение фильтра частиц, как инструмента для моделирования систем случайной структуры, представляет собой эффективное решение для оценивания параметров системы и её состояния даже при сложных вероятностных распределениях. Дальнейшие исследования могут быть направлены на оптимизацию вычислительной эффективности алгоритма.

Список литературы:

1. Жданов Э. Р., Маликов Р. Ф., Хисматуллин Р. К. Компьютерное моделирование физических явлений и процессов методом Монте-Карло: учеб. Пособие. – 2005.

2. Казаков И.Е. Статистическая динамика систем с переменной структурой. – М.: Наука, 1977.

3. Парахневич А. В., Солонар А. С., Горшков С. А. Фильтрация посредством выборки весовых коэффициентов. Обобщенный фильтр частиц (particle filter) // Доклады БГУИР. 2012. №3 (65).

4. Степанов О.А. Основы теории оценивания с приложениями к задачам обработки навигационной информации. Ч.2. Введение в теорию фильтрации. ГНЦ РФ ОАО «Концерн «ЦНИИ «Электроприбор», 2012. – 417с.