УДК 62-519

ПРОГРАММНЫЙ АЛГОРИТМ АППРОКСИМАЦИИ СЛОЖНОГО ДИНАМИЧЕСКОГО ЗВЕНА АПЕРИОДИЧЕСКИМ ЗВЕНОМ ВТОРОГО ПОРЯДКА ПРИ ПОМОЩИ МЕТОДА РОЯ ЧАСТИЦ

[1]Пиотровский Д.Л., [2]Куколев А.А., [2]Подгорный С.А.

[1]ФГБОУ ВО «Московский государственный университет технологий и управления имени К.Г. Разумовского», Москва, e-mail: piotrovsky2005@yandex.ru;

[2]ФГБОУ ВО «Кубанский государственный технологический университет», Краснодар, e-mail: sashanius@yandex.ru, saptich@rambler.ru

К настоящему времени, в связи с повсеместной интеграцией электронно-вычислительных и сетевых технологий взаимодействия, возросла тенденция к использованию методов, позволяющих реализовать автоматизированный поиск оптимального решения в соответствии с заданным набором ограничений, регламентируемых поставленной задачей. Одним из подобных методов является предложенный Джеймсом Кеннеди и Расселом Эберхартом алгоритм роя частиц. Концепция алгоритма была частично основана на результатах исследования поведения скоплений животных (косяков рыб, стай птиц и т.д.). Данная статья предлагает к рассмотрению программную реализацию метода роя частиц в задаче определения эквивалентного аппроксимирующего звена, соответствующего сложному звену с заранее известной переходной характеристикой. Статья предоставляет к рассмотрению некоторые аспекты реализации метода в программной среде Microsoft Visual Studio. За основу сложного звена взят контур подготовки топлива судового главного малооборотного двигателя с электронным управлением подачей топлива MAN B&W 6S90 ME-C. Описание структуры звена предоставлено в рамках необходимого для рассмотрения основного вопроса исследования объема. Статья также предоставляет описание некоторых полученных после моделирования результатов и соответствующих заключений.

Ключевые слова: обработка информации, оптимизация, метод роя частиц, управление, Visual C#, Microsoft Visual Studio

PARTICLE SWARM OPTIMIZATION METHOD SOFTWARE ALGORYTHM FOR COMPLEX CONTROL SYSTEM DYNAMIC LINK APPROXIMATION WITH SECOND ORDER APERIODIC LINK

[1]Piotrovskiy D.L., [2]Kukolev A.A., [2]Podgornyy S.A.

[1]K.G. Razumovsky Moscow State University of Technologies and Management, Moscow, e-mail: piotrovsky2005@yandex.ru;

[2]Kuban State Technological University, Krasnodar, e-mail: sashanius@yandex.ru, saptich@rambler.ru

Nowadays due to constant network and computer- based technologies interaction there have appeared a trend to use the data processing methods for automatic optimal solution determination in accordance with pre-defined limits set, being contained by the problem in question. One of the these methods is the suggested by Kennedy and Ebethart particle swarm optimization method. The core of the method was partially based on animal behavior exploration results. This article gives an overview of the particle swarm optimization method C# program code for the problem of complicated automatic control system link optimization with the 2nd order aperiodic link, being described with the known transfer chart. The article represents several Microsoft Visual Studio C# code aspects for the particle warm optimization method implementation. Complicated link model is here known to be based on ship 2-stroke diesel main engine MAN B&W 6S90 ME-C with electronically controlled fuel injection. The required link description is described as well in accordance with idea of the required for basic algorithm understanding information. The article describes some modeling results and conclusions as well.

Keywords: data processing, optimization, particle swarm optimization, control, Visual C#, Microsoft Visual Studio

Методы оптимизации и линейного программирования берут свое начало в 1820 г., когда Фурье предложил метод направленного перебора смежных вершин в направлении возрастания целевой функции – симплекс-метод, ставший на долгое время основным при решении задач линейного программирования [1]. Позднее в 1947 г. метод был расширен и дополнен американским ученым Джоржем Данцигом. Класс экстремальных задач, определяемых линейным функционалом на множестве, задаваемом линейными ограничениями, относят к 1930-м гг., когда появились первые разработки по решению задач линейного программирования исследователей Джона фон Неймана, Л.В. Канторовича [2, 3]. Тем же Канторовичем совместно с М.К. Гавуриным в 1949 г. был разработан метод потенциалов, являвшийся модификацией симплекс-метода решения задачи линейного программирования применительно к транспортной задаче. В последующих работах Л.В. Канторович, В.С. Немчинов, В.В. Новожилов, А.Л. Лурье, А. Брудно развили математическую теорию линейного и нелинейного

программирования, в том числе развивая приложения к этой теории. В целом, в связи с развитием электронно-вычислительных средств и компьютерной техники, начиная с 1955 г. опубликовано множество работ как в области нелинейного программирования, так и в области квадратичного программирования (работы Баранкина, Дорфмана и пр.) [4, 5].

Актуальным вопросом при исследовании сложных систем зачастую становится задача упрощения сложных звеньев более простыми. В частности, сложный объект управления с большим количеством обратных связей – судовой главный двигатель – зачастую необходимо упростить или заменить более простым звеном.

Авторами была сформулирована цель исследования – рассмотреть и проанализировать возможность программной реализации метода роя частиц (МРЧ) в задаче аппроксимации сложного динамического звена простым апериодическим звеном второго порядка.

Материалы и методы исследования

Для рассмотрения передаточной функции топливной аппаратуры необходимо рассмотреть принцип произведения впрыска в судовом главном двигателе с электронным управлением подачей топлива типа MAN B&W xSxx ME-C. Принцип действия топливной аппаратуры раскрывает рис. 1 [6]. На последних моделях двигателей MAN B&W серии ME-C в качестве гидравлического блока цилиндра используется гидравлический блок HCU (Hydraulic cylinder unit), состоящий из топливного насоса высокого давления (ТНВД), исполнительного механизма выпускного клапана, гидравлического аккумулятора, лубрикатора цилиндра, электрогидравлического клапана корректировки подачи топлива, блока распределения гидравлики, трубопроводов, а также датчиков положения механизмов.

Таким образом, в данном случае процесс подачи топлива можно представить в виде нескольких систем уравнений: уравнений работы электромеханической части электромагнитного клапана корректировки, уравнений работы гидравлической части электромагнитного клапана корректировки, уравнений работы распределителя гидравлики с трубопроводами, а также уравнений ТНВД с форсункой (рис. 2).

Положим эквивалентную передаточную функцию сложного звена равной

$$W_{\text{топл}}(s) = \frac{521.2}{(0.1s+500)\left(5\cdot10^{-2}s^2+s+1.86\right)\left(0.2\cdot10^{-12}s+3.61\cdot10^{-6}\right)\left(3.24s^2+55.56s+10^4\right)\left(1.8\cdot10^{-12}s+3\cdot10^{-7}\right)\left(32s^2+14.1s+4\cdot10^6\right)(s+1)}.$$

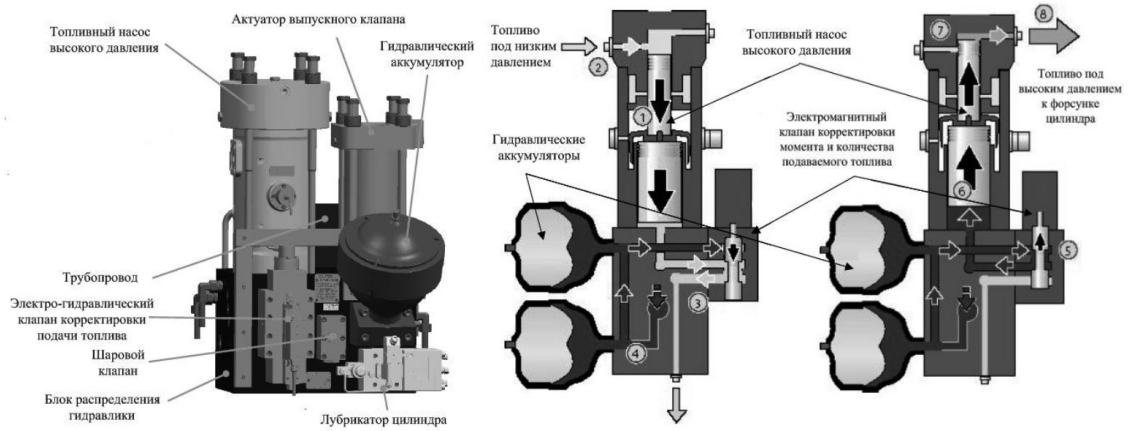

Рис. 1. Устройство и принцип действия блока HCU двигателя MAN B&W 6S90 ME-C

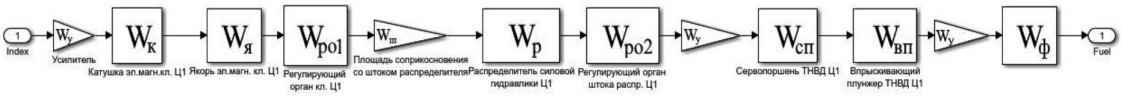

Рис. 2. Структурная схема контура подготовки топлива и клапана FIVA

Тогда целевой задачей является аппроксимация звена $W_{\text{топл}}(s)$ звеном:

$$W_{\text{экв}}(s) = \frac{k_1}{k_2 s^2 + k_3 s + k_4}.$$

Алгоритм роя частиц декларирует в своем составе рой одинаковых с точки зрения выполняемой функции частиц, обладающих положением и скоростью в текущий момент времени. Такими же свойствами обладает и весь рой в целом. Решение задачи оптимизации сводится к поиску оптимального положения частицы в n-мерном пространстве поиска решения посредством определения текущей ошибки положения частицы, вычисления скорости частицы, а также соответствующих свойств роя.

Корректировка скоростей и положений каждой из частиц в изначальной конфигурации производилась в соответствии с формулами [7]:

$$v_n(t+1) = \left(v_n(t) + c_1 r_1 \left(\vec{p}_n(t) - \vec{x}_n(t)\right) + c_2 r_2 \left(\vec{g}(t) - \vec{x}_n(t)\right)\right);$$

$$\vec{x}_n(t+1) = \left(x_n(t) + v_n(t+1)\right),$$

где $v_n(t+1)$ – скорость частицы n в момент времени $(t+1)$, $v_n(t)$ – скорость частицы в текущий момент времени t, $c_1 = const$ – когнитивная весовая доля, $r_1 = const$ – случайная переменная в диапазоне [0;1], $\vec{p}_n(t)$ – векторная величина, описывающая лучшее положение частицы, найденное на данный момент, $\vec{x}_n(t)$ – текущая позиция частицы, $\vec{g}(t)$ – лучшая известная позиция, найденная для любой из частиц в рое, $\vec{x}_n(t+1)$ – положение частицы n в момент времени $(t+1)$.

Текущая задача была решена на основе приложения, написанного на языке Visual C#. Для этого целесообразно рассмотреть алгоритм решения поставленной задачи при помощи рис. 3. Алгоритм берет свое начало с процедуры определения заданных оператором параметров расчета. Среди них – количество эпох расчета для каждого шага времени t, количество частиц n, размерность пространства поиска решений, а также максимально допустимые значения искомых параметров. Следом производится расчет начального положения частиц и их скоростей, исходя из значений которых определяется текущая ошибка расчета:

$$E = \prod_{t_0=0}^{t} E\left(y(t), z(t)\right),$$

где $y(t)$ и $z(t)$ – временные зависимости заданной и искомой временной характеристики.

Далее инициализируется основной цикл обращения к каждой частице, содержащий в своем составе еще несколько циклов и основные операции, а именно: определение скоростей и положений всех частиц в последующие моменты времени, определение ошибок всех рассчитанных положений и проверки на превышение максимально допустимых текущими параметрами значений. Рассмотрим реализацию поставленной задачи подробнее.

Рис. 3. Алгоритм решения задачи поиска минимума функции по методу роя частиц

В основу работы приложения заложен метод, реализующий инициализацию роя частиц, описание их свойств, расчет текущих скоростей и положений и определение ошибки положения каждой частицы и всего роя, который представляет собой код [8, 9]:

```
private void button2_Click(object sender, EventArgs e)
     {
          Random ran = null;
          ran = new Random(0);
          try
          {
              Particle[] swarm = new Particle[numberParticles];
              double bestGlobalFitness = double.MaxValue;
              double minV = -1.0 * maxX;
               double maxV = maxX;
               for (int i = 0; i < swarm.Length; ++i) 30                {
                   double[] randomPosition = new double[Dim];
                   for (int j = 0; j < randomPosition.Length; ++j)
                   {
                       double lo = minX;
                       double hi = maxX;
                       randomPosition[j] = (hi - lo) * ran.NextDouble() + lo;
                   }
                   double fitness = ObjectiveFunction(randomPosition);
                   double[] randomVelocity = new double[Dim];
           for (int j = 0; j < randomVelocity.Length; ++j)
                   {
                       double lo = -1.0 * Math.Abs(maxX - minX);
                       double hi = Math.Abs(maxX - minX);
                       randomVelocity[j] = (hi - lo) * ran.NextDouble() + lo;
                   }
       swarm[i] = new Particle(randomPosition, fitness, randomVelocity, randomPosition, fitness);
            if (swarm[i].fitness < bestGlobalFitness)
                   {
                       bestGlobalFitness = swarm[i].fitness;
                       swarm[i].position.CopyTo(bestGlobalPosition, 0);
                   }
               }
               double w = 0.729;
               double c1 = 1.49445;
               double c2 = 1.49445;
               double r1, r2;
               while (iteration < numberIterations)
               {
                   ++iteration;
                   double[] newVelocity = new double[Dim];
                   double[] newPosition = new double[Dim];
                   double newFitness;
                   for (int i = 0; i < swarm.Length; ++i)
                   {
                       Particle currP = swarm[i];
                       for (int j = 0; j < currP.velocity.Length; ++j)
                       {
                           r1 = ran.NextDouble();
                           r2 = ran.NextDouble();
                           newVelocity[j] = (w * currP.velocity[j]) +
                             (c1 * r1 * (currP.bestPosition[j] - currP.position[j])) +
                             (c2 * r2 * (bestGlobalPosition[j] - currP.position[j]));
                           if (newVelocity[j] < minV)
                               newVelocity[j] = minV;
                           else if (newVelocity[j] > maxV)
                               newVelocity[j] = maxV;
                       }
                       newVelocity.CopyTo(currP.velocity, 0);
                       for (int j = 0; j < currP.position.Length; ++j)
                       {
                           newPosition[j] = currP.position[j] + newVelocity[j];
                           if (newPosition[j] < minX)
                               newPosition[j] = minX;
                           else if (newPosition[j] > maxX)
                               newPosition[j] = maxX;
                       }
                       newPosition.CopyTo(currP.position, 0);
```

```
69                          newFitness = ObjectiveFunction(newPosition);
70                          currP.fitness = newFitness;
71                         if (newFitness < currP.bestFitness)
72                          {
73                              newPosition.CopyTo(currP.bestPosition, 0);
74                              currP.bestFitness = newFitness;
75                          }
76                          if (newFitness < bestGlobalFitness)
77                          {
78                              newPosition.CopyTo(bestGlobalPosition, 0);
79                              bestGlobalFitness = newFitness;
80                          }
81                      }
82                  }
83              }
84              catch (Exception ex)
85              {
86              }
87          }
```

Цикл начинается с присваивания случайного значения переменной `ran`, необходимой для расчета случайного значения коэффициентов r_1 и r_2 в формуле определения последующей скорости частицы $v_n(t+1)$. Строка 5 инициализирует цикл `try`, в котором и осуществляется расчет всех параметров частиц на всех итерациях состояний всех моментов времени до появления некоторой ошибки расчета, реализуемой функцией `catch (Exception ex)`. Инициализация роя частиц возложена на код строки 7, в котором при помощи внешнего класса `Particle` инициализируется переменная `swarm`, передающая в конструктор одинединственный заданный оператором параметр – `numberParticles` – количество частиц роя. Цикл строки 13 определяет положение текущей частицы в текущий момент времени. Строка 19 определяет ошибку текущего положения текущей частицы на каждой итерации времени моделирования посредством вызова внешнего метода `ObjectiveFunction`:

```
    static double ObjectiveFunction(double[] k_T_b)
     {
                int k = 0;
                double[] cur_error = new double[101];
                double result = 1;
                for (int n = 0; n < cur_error.Length; ++n)
                {
                    cur_error[n] = 1;
                }
                for (double i = 0.1; i < 10.1; i=i+.1)
                {
cur_error[k] = (2.46 * Math.Pow(10, -15) * Math.Exp(-5 * Math.Pow(10, -3)) * i + 7.68 *
Math.Pow(10, -46) * Math.Exp(-1.81 * Math.Pow(10, 7)) - 1.62 * Math.Pow(10, -27) *
Math.Exp(-1.67 * Math.Pow(10, 5) * i) - 26.6 * Math.Exp(-1 * i) + 12.9 + 13.7 * Math.Exp(-10 * i) *
Math.Cosh(7.92 * i) + 13.8 * Math.Exp(-10 * i) * Math.Sinh(7.92 * i) - 7.11 * Math.Pow(10, -4) *
Math.Exp(-8.57 * i) * Math.Cos(54.9 * i) + 2.84 * Math.Pow(10, -3) * Math.Exp(-8.57 * i) *
Math.Sin(54.9 * i) - 8.92 * Math.Pow(10, -9) * Math.Cos(354 * i) -2.74 * Math.Pow(10, -7) *
Math.Exp(-221 * i) * Math.Sin(354 * i)) - (1 * (k_T_b[0] - k_T_b[0] * Math.Exp(k_T_b[1] * i) *
Math.Cos(k_T_b[2] * i) - 0.115 * Math.Exp(k_T_b[1] * i) * Math.Sin(k_T_b[2] * i)));
                k = k + 1;
                }
                for (int j = 0; j < cur_error.Length; ++j)
                {
                    result = Math.Abs(result * cur_error[j]);
                }
                return result;
     }
```

Цикл строки 21 определяет массив значений скоростей каждой частицы. Строка 21 инициализирует частицу `swarm` с определенными ранее параметрами. Условие `if` строки 28 проверяет соответствие значения текущей ошибки положения частицы заданному. Далее инициализируются переменные расчета новых положения и скорости частицы c_1, c_2, r_1, r_2 и весовой коэффициент инерции w. Последующие циклы определяют новое положение и скорость текущей частицы и новую ошибку ее положения, в случае нового минимального значения которой последнее обновляется и ему присваивается значение текущей ошибки расчета.

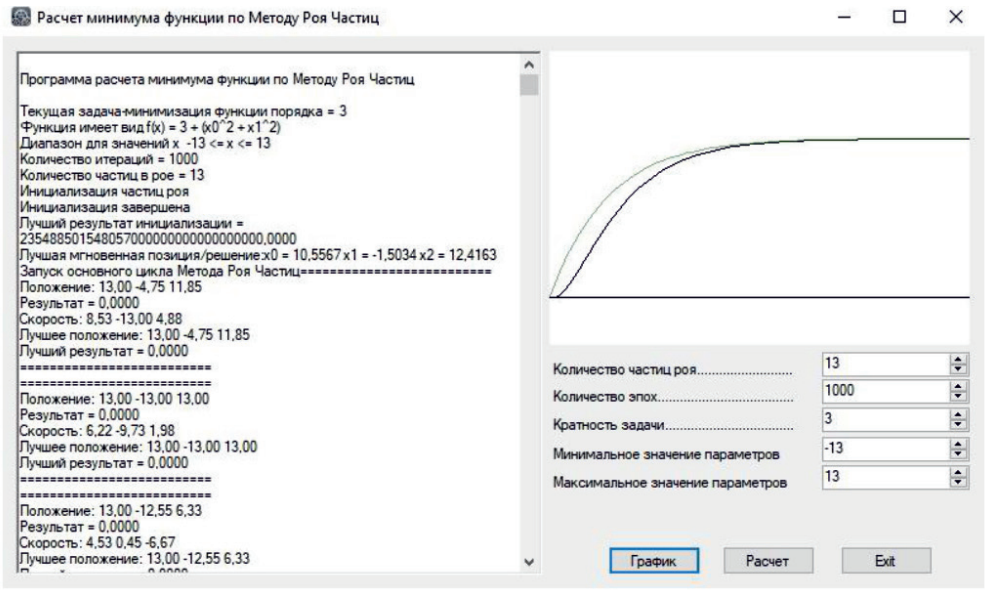

Рис. 4. Интерфейс программного обеспечения для поиска минимума по МРЧ

Результаты исследования и их обсуждение

Задачей алгоритма служит определение коэффициентов α, β и γ аппроксимирующего звена в соответствии с его временной характеристикой:

$$z(t) = \alpha - \alpha \cdot e^{\gamma t} \cdot \cos(\beta t) - 0.114 \cdot e^{\gamma t} \cdot \sin(\beta t),$$

полученной посредством обратного преобразования Фурье:

$$z(t) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} I(j\omega) \cdot e^{j\omega t} d\omega.$$

Функция `ObjectiveFunction` принимает в качестве параметра лишь один аргумент – массив значений переменных α, β и γ `double[] k_T_b`. В этом массиве содержатся все текущие значения решений поставленной задачи аппроксимации. Цикл `for (double i = 0.1; i < 10.1; i = i+.1)` производит определение текущей ошибки *E* итерации `k` посредством определения разности значения переходной характеристики эталонного звена и искомого аппроксимирующего звена, а также заполняет массив значений `cur_error[k]`. Переменная `result` определяет результирующую ошибку расчета на всех итерациях, возвращая рассчитанное значение в основной цикл `try`.

Итогом расчета является оптимальное положение частицы в трехмерном пространстве поиска решения. В данном случае аппроксимация указанной сложной передаточной функции реализуется массивом тысячи эпох перемещения частиц посредством звена вида (рис. 4):

$$W_{\text{топл}}(s) = \frac{96.2 \cdot s + 85.8}{10 \cdot s^2 + 16 \cdot s + 6.6},$$

которому соответствует временная характеристика вида

$$z(t) = 12{,}9 - 12{,}9 \cdot e^{0{,}12t} \cdot \cos(-0{,}8t) - -0.114 \cdot e^{0{,}12t} \cdot \sin(-0{,}8t).$$

Заключение

Рассмотренная структура предназначена для аппроксимации сложных звеньев и подтверждает допустимую возможность определения аппроксимирующего апериодического звена второго порядка для сложного динамического звена. Как видно из предоставленного результата, данный метод не лишен недостатков. Стоит отметить уже обозначенный выше недостаток – вероятность определения неистинного минимума, что косвенно подтверждается и в данном случае. Начало переходного процесса полученного звена недостаточно корректно соответствует эталонному сложному звену. Однако здесь следует сделать замечание относительно возможностей описанного алгоритма. Так как поиск решения ведется в трехмерном пространстве, то неизменным остается коэффициент -0,114 при отрицательной части полинома. По этой причине очевидна дальнейшая необходимость мо-

дификации описанной структуры с целью минимизации ошибки расчета и поиска решения на пространстве четырех измерений.

Список литературы

1. Хемди А. Таха. Глава 3. Симплекс-метод. М.: «Вильямс», 2007. 912 с.

2. John Von Neumann The computer and the brain: The Silliman Memorial Lectures Series, 2012. 136 p.

3. Канторович Л.В. Математические методы организации планирования производства // Издание Ленинградского государственного университета. М.: Ленинград, 1939. 760 с.

4. Гальперин В.М. Математическое, или «линейное», программирование: нематематическое представление // Вехи экономической мысли. Т. 2. Теория фирмы / Под ред. В.М. Гальперина. СПб.: Экономическая школа, 2000. 534 с.

5. Grechuk B., Molyboha A., Zabarankin M. Mean-deviation analysis in theory of choise. Risk Analysis: An International Journal. 2012. Т. 32. № 8. P. 1277–1292.

6. Дайнего Ю.Г. Эксплуатация судовых энергетических установок, механизмов и систем. Практические советы и рекомендации М.: «Моркнига», 2018. 340 с.

7. Xinchao Z. A perturbed particle swarm algorithm for numerical optimization. Applied Soft Computing. 2010. Т. 10. № 1. P. 11–124.

8. Маккаффри Д. Метод роя частиц // MSDN Magazine Issues-Artificial intelligence. 2011. № 8. [Electronic resource]. URL: https://docs.microsoft.com/ru-ru/archive/msdn-magazine/2011/august/artificial-intelligence-particle-swarm-optimization (date of access: 21.04.2021).

9. Пиотровский Д.Л., Подгорный С.А., Куколев А.А. Программное описание алгоритма работы ПИД-регулятора на языке C# для применения в замкнутой системе автоматического управления // Сборник научных трудов НГТУ. 2020. № 1–2 (97). С. 40–54.