УДК 004.43+519.6+519.622

ОПТИМАЛЬНИЙ ВИБІР МОВИ ПРОГРАМУВАННЯ ДЛЯ РОЗРОБКИ ПРОГРАМ МОДЕЛЮВАННЯ ДИНАМІЧНИХ СИСТЕМ

А. Харів, І. Хвищун, М. Баран

Львівський національний університет імені Івана Франка
вул. Ген. Тарнавського, 107, 79017 Львів, Україна
khvyschun@electronics.wups.lviv.ua

Значну кількість задач моделювання динамічних режимів технічних пристроїв зводять до розв'язання систем лінійних алгебричних рівнянь. Схожі системи найчастіше розв'язують методом Гауса або його модифікаціями. У статті подано результати дослідження залежності швидкодії алгоритму методу Гауса від мови програмування, якою його реалізовано.

Ключові слова: моделювання, чисельний метод, швидкодія, метод Гауса, мова програмування.

Важливою сферою застосування комп'ютерів є автоматизоване проектування технічних пристроїв та систем. Однією із найскладніших та найвідповідальніших проектних операцій є математичне моделювання пристрою, що проектується. Поруч із алгоритмами моделювання вибір мови програмування, якою написані моделюючі програми, має суттєвий вплив на швидкодію процесу моделювання.

Відомо, що процес математичного моделювання багатьох типів динамічних систем, зокрема аналогових електронних схем, налічує три основні етапи [1]:

1. формування математичної моделі (ММ) у вигляді системи звичайних диференційних рівнянь або у канонічній формі Коші: $\dot{x} = \frac{dx}{dt} = f(x,t)$, або у неявній формі: $F(x, \dot{x}, t) = 0$, де x, f, F – вектор-функції;
2. розв'язання цієї системи чисельними методами;
3. аналіз результатів моделювання та прийняття рішення щодо відповідності характеристик пристрою технічному завданню на його проектування або вибору способу оптимізації його параметрів чи структури.

З погляду затрат ресурсів комп'ютера найскладнішим є другий етап моделювання. На цьому етапі, для чисельного інтегрування систем диференційних рівнянь, зазвичай застосовують неявні методи прогнозу-корекції, оскільки ММ, як правило, є жорсткими. В основу роботи цих методів покладено алгебризацію рівнянь ММ, тобто перетворення їх у систему нелінійних алгебричних рівнянь, яку розв'язують ітераційним методом Ньютона [1]. На кожній ітерації цю нелінійну систему лінеаризують шляхом зведення її

до системи лінійних алгебричних рівнянь (СЛАР) типу $A * x = b$, де x – вектор розв'язку СЛАР, A розріджена або заповнена матриця із дійсними елементами.

Розв'язують такі системи методом Гауса або його модифікаціями.

У статті подано результати тестування алгоритму Гауса [1] для розв'язання СЛАР із заповненою матрицею розміру (200×200), елементами якої є дійсні числа, сформовані за алгоритмом Гільберта:

$$A_{i,j}^{N} = \frac{1}{i + j - 1}, \quad 1 \le i \le N, \quad 1 \le j \le N, \text{ де } N \text{ – розмірність СЛАР.}$$

Відомою особливістю матриці Гільберта є її погана обумовленість.

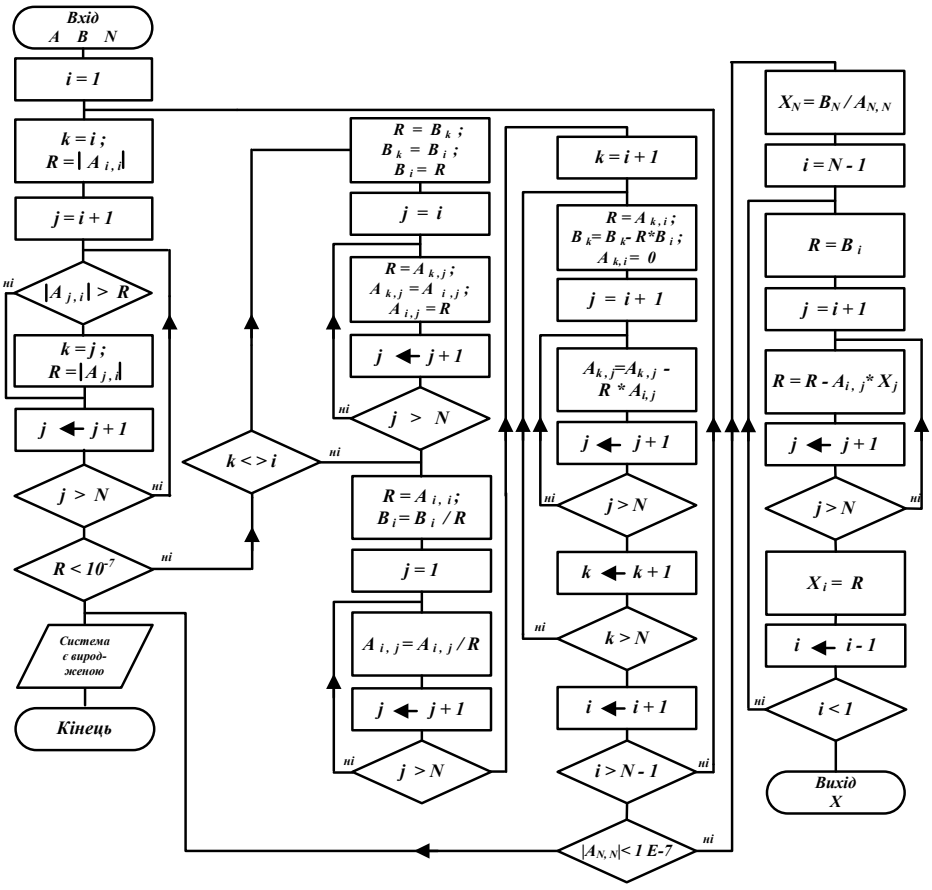

Рис. 1. Алгоритм методу Гауса

Розв'язки СЛАР на основі матриці Гільберта знаходили з допомогою програм, написаних мовами програмування, якими найчастіше програмують системи моделювання: Fortran [2], Object Pascal [3], C++ [4], Java [5], C# [6]. Масив значень елементів матриці та вектора правої частини СЛАР формували або як статичний (Static) (формується на етапі компілювання програми), або як динамічний (Dynamic) (формується на етапі виконання програми). Обчислення проводили на комп'ютерах із процесорами різних поколінь. Зазначимо, що у мові C# статичних масивів немає.

```
    static public void Gauss (ref double[,] A, ref double[] B, int N,
ref double[] X)
     {
      int i, j, k;
      double R;
      for (i = 1; i <= N - 1; i++)
       {
        k = i;
        R = Math.Abs(A[i, i]);
        for (j = i + 1; j <= N; j++)
         if (Math.Abs(A[j, i]) >= R)
          {
           k = j;
           R = Math.Abs(A[j, i]);
          }
         if (k != i)
          {
           R = B[k];
           B[k] = B[i];
           B[i] = R;
           for (j = i; j < N; j++)
            {
             R = A[k, j];
             A[k, j] = A[i, j];
             A[i, j] = R;
            }
           }
                    R = A[i, i];
                    B[i] = B[i] / R;
                    for (j = 1; j <= N; j++)
                        A[i, j] = A[i, j] / R;
                    for (k = i + 1; k <= N; k++)
                    {
                        R = A[k, i];
                        B[k] = B[k] - R * B[i];
                        A[k, i] = 0;
                        for (j = i + 1; j <= N; j++)
                            A[k, j] = A[k, j] - R * A[i, j];
                    }
                }
              X[N] = B[N] / A[N, N];
              for (i = N - 1; i >= 1; i--)
              {
                  R = B[i];
                  for (j = i + 1; j <= N; j++)
                      R = R - A[i, j] * X[j];
                  X[i] = R;
              }
          }
      }
```

Рис. 2. Реалізація алгоритму методу Гауса мовою C#

Наприклад, рис. 2 свідчить про реалізацію алгоритму Гауса у вигляді методу, написаного мовою програмування C#. Оскільки для його тестування ми обрали матрицю Гільберта, яка не є виродженою, то у програмі немає операторів перевірки матриці A на виродження, хоча в алгоритмі, що на рис. 1, вони наявні.

Матрицю *A* і вектор *b* формували так:

```
for (int i = 1; i <= N; i++)
   for (int j = 1; j <= N;j++ )
    {
      a[i][j] = 1.0 / (i + j + 1);
      b[j] = (double)j;
}
```

Рис. 3. Формування СЛАР мовою C#

З метою оцінки швидкодії роботи програм для різних мов програмування використано різні таймери, однак ідея однакова – з точністю 10^{-3}с проводили замір системного часу на початку формування СЛАР і після завершення її розв'язування.

Для Visual C++, Delphi, GCC, Borland C++ Builder використано покази таймера **`GetTickCount()`**, що повертає системний час в мілісекундах.

Для C# використано структуру **`DateTime.Now`**, що повертає поточну дату й час.

В Java використано аналог **`GetTickCount()`** – **`System.currentTimeMillis()`**, який повертає системний час в мілісекундах.

У мові Fortran використано функцію **`date_and_time()`** з параметром **`time`**, що повертає час з точністю до мілісекунд.

Нижче, у таблиці наведено результати цих досліджень. У всіх дослідах використано консольні варіанти програм. Час розв'язання СЛАР подано у мілісекундах.

Таблиця

Тип процесора / Мова програмування	1	2	3	4	5	6	7
Fortran	**1162**	**1240**	**1306**	**1554**	**2640**	**3545**	**4000**
Delphi (Static)	**5557**	**5148**	**5647**	**8143**	**4047**	**6314**	**9578**
Delphi (Dynamic)	**3633**	**7067**	**7457**	**6209**	**8532**	**13969**	**5157**
C ++ Builder (Dynamic) Non optimized	**5090**	–	–	–	–	**16753**	–
C ++ Builder (Dynamic)	**3157**	–	–	–	–	**6209**	–
VC++ (Static)	**869**	**811**	**858**	**1077**	**2016**	–	**1156**
VC++ (Dynamic)	**883**	**827**	**889**	**1045**	**2031**	–	**1187**
GnuCC	**637**	**1061**	**936**	**1061**	**2094**	**3144**	**1312**
Java (Dynamic)	–	**1477**	**1872**	**2055**	**2875**	–	**4406**
C# (Dynamic)	**3518**	**4542**	**4851**	**4910**	**7406**	**10704**	**4406**

У таблиці взято таку нумерацію типів процесора:

1. Intel Core 2 Duo E8400 3,00 Ghz.
2. Intel Core 2 Duo P'8600 2,4 Ghz.
3. Intel Core 2 Duo T8100 2,1 Ghz.
4. Intel Core 2 Duo E4400 2,0 Ghz.
5. AMD Turion X2 1,8 Ghz.
6. AMD Athlon 2500 XP+ 1,8 Ghz.
7. Intel Pentium 4 2,4 Ghz.

На рис. 4. подано графічне представлення усереднених результатів тестування у мілісекундах.

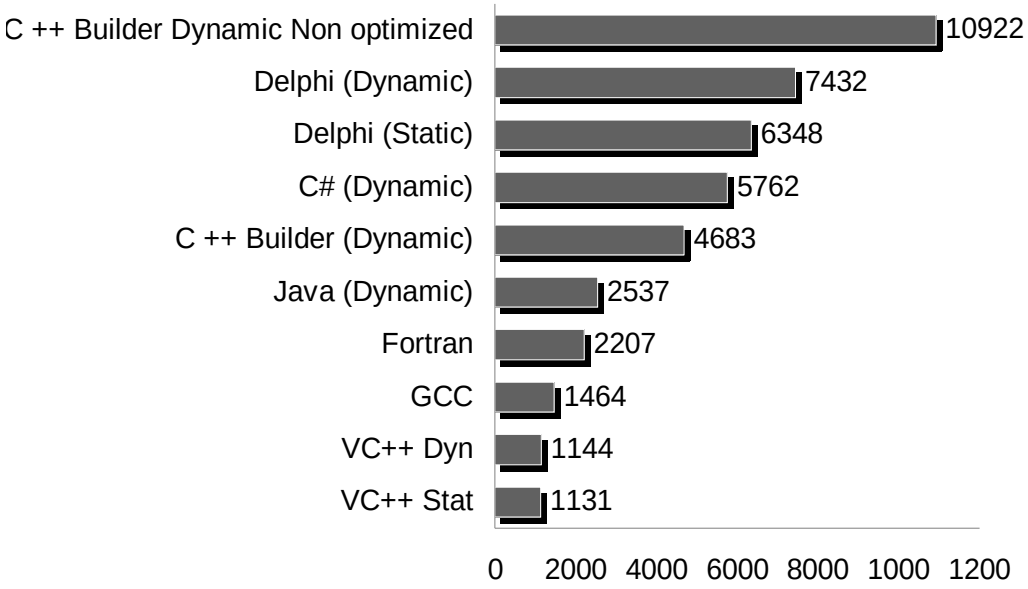

Рис. 4. Усереднені результати

Дослідження засвідчують, що тип процесора є важливим, але не основним чинником впливу на швидкодію програм математичного моделювання. Суттєвішим є вибір мови реалізації програми, оскільки оптимальне формування двійкового коду дає значно більший виграш у швидкодії, ніж збільшення тактової частоти процесора чи кількості його ядер. Найкращі результати показала мова C++, а також Fortran і Java, причому результати Java суттєво залежать від типу процесора, а C++ навпаки показує однаково хороші результати незалежно від процесора.

1. *Хвищун І.О.* Програмування і математичне моделювання: Підручник. К.: Ін Юре, 2007. 544 с.
2. *Алгазин С. Д., Кондратьев В. В.* Программирование на Visual Fortran. Диалог-МИФИ, 2008. 472 с.
3. *Фленов М. Е.* Библия Delphi. СПб.: БХВ-Петербург, 2004. 880 с.
4. *Страуструп Б.* Язык программирования C++. Бином. 2004. 1104 с.
5. *Шилдт Г.* Полный справочник по Java, Вильямс, 2007. 1040 с.
6. *Неш Т.* C# 2008: ускоренный курс для профессионалов. Пер. с англ. М.: ООО “М.Д. Вильямс”, 2008. 576 с.

OPTIMAL SELECTION OF PROGRAMMING LANGUAGE FOR DEVELOPMENT OF APPLICATIONS FOR DYNAMICAL SYSTEMS MODELLING

A. Khariv, I. Khvyshchun, N. Baran

Ivan Franko National University of L'viv
Tarnavsky Str., 107, UA–79017 Lviv, Ukraine
khvyschun@electronics.wups.lviv.ua

A considerable number of problems of simulation technical devices dynamics are being solved as linear algebraic equation systems. Such systems are often being solved using Gauss method or its modifications. The issue presents the results of Gauss algorithm performance depending from the programming language of its implementation.

Key words: simulation, numerical method, performance, Gauss method, programming language.

ОПТИМАЛЬНЫЙ ВЫБОР ЯЗЫКА ПРОГРАММИРОВАНИЯ ДЛЯ РАЗРАБОТКИ ПРОГРАММ МОДЕЛИРОВАНИЯ ДИНАМИЧЕСКИХ СИСТЕМ

А. Харив, И. Хвищун, Н. Баран

Львовский национальный университет имени Ивана Франко
ул. Ген. Тарнавского, 107, 79017 Львов, Украина
khvyschun@electronics.wups.lviv.ua

Значительное количество задач моделирования динамических режимов технических устройств сводят к решению систем линейных алгебраических уравнений. Подобные системы чаще всего решают методом Гаусса или его модификациями. В работе даны результаты исследования зависимости быстродействия алгоритма метода Гаусса от языка программирования, которым он реализован.

Ключевые слова: моделирование, численный метод, быстродействие, метод Гаусса, язык программирования.

Стаття надійшла до редколегії 21.04.2009
Прийнята до друку 30.06.2009