Design and Implementation of an iterative optimization algorithm based on the C language

YaoCe

Tianjin Lighe Industry Vocational Technical College 300350

Keywords: C language; iterative algorithm; Optimization

Abstract: An iterative optimization algorithm based on C language is designed and implemented in this paper, the algorithm is based on the traditional algorithm, and check constraint is added to ensure the correctness of the iterative process. The simulation software is regarded as the basis for computer simulation experiment, experimental results show that the algorithm converges, and have more efficiency, so as to meet the needs of practical application.

1 Introduction

In the computer environment, for considerations of efficiency, the data of communications and storage are required to be calculated by people easily, in case to effect the efficiency. In addition, with the wide application of computer and communication networks, the efficient processing of information has been paid widespread attention. Efficient processing algorithm of information has not only limited to political, military, and diplomatic and other fields, it has been closely linked with people's daily life [1-3].

However, with the increasing rate of modern computer systems and growth of information processing tools, some traditional information processing algorithms are no longer valid. And the processing efficiency is getting lower, so even some common data processing measures are adopted, such as: access iterative control and other methods, efficiency of data processing still cannot meet the requirements [4-5]. Therefore, the algorithm needs to be further studied to ensure efficiency of processing. This paper presents an improved iterative algorithm, and implemented with the C language.

2 principle of the iterative optimization algorithm

Iterative algorithm is the most common algorithm in data processing, the current iterative algorithm is optimized in the paper to benefit processing data, and a learning mechanism is introduced, the principle works as follows:

The principle of iterative learning algorithm is: When the calculation is for k-th iteration, before the arrival of next iteration, compensation data Δi_q^k is found by table look-up module, and transferred to the main control algorithm, so as to compensate the iterative fluctuation, and Δi_q^k is stored in the memory; then by predicting the difference of iterative fluctuation Δf_k and expected fluctuation Δf_q, i.e., $e_k(t) = \Delta f_k - \Delta f_q$, and $\Delta i_q^k(t)$ in memory 3, after the learning process, the learning rules is able to obtain a new compensation data $\Delta i_q^{k+1}(t)$ as the k + 1-th compensated value of iteration calculation, and then start a new iteration operation. Feed - forward compensation effect is achieved.

The learning law of iterative learning algorithm can be designed as follows:

$$\Delta i_q^{k+1}(t) = \Delta i_q^k(t) + \Gamma_p e_k(t) + \Gamma_i \int_0^t e_k(\tau)d\tau + \Gamma_D \frac{de_k(t)}{dt} \quad (1)$$

After repeated iterative learning, the final modulated result adjusts the thrust volatility close to zero as possible.

3 The implementation of algorithm based on C language

The code of main members' function in optimized iterative learning algorithm is as follows:

1. Constructing the function, according to the number of iterations to assign keyBytes to key128, key192, key256.

```
Aes(int keysize, unsigned char* keyBytes)
    {   if(keysize==16)
        {   Nb=4;Nk=4;Nr=10;                          // Set iteration count
          memcpy(key128,keyBytes,keysize);
          KeyExpansion128();                  // Initialization
        }
        else if(keysize==24)
        {   Nb=4;Nk=6;Nr=12;                          // Set iteration count
           memcpy(key192,keyBytes,keysize);
           KeyExpansion192();                 // Iterative extension, the initialization must be done in advance
        }
        else if(keysize==32)
        { Nb=4;Nk=8;Nr=14;                            // Set iteration count
          memcpy(key256,keyBytes,keysize);
          KeyExpansion256();                  // Initialization
        }
    }
```

The functions in iterative learning process:

```
    KeyExpansion256()                                    // 256 bit storage expansion
    {    memset(w256,0,16*15);
         for(int i=0;i<Nk;i++)                  // Copy iteration data of seed
         {     w256[4*i+0] =   key256[4*i];w256[4*i+1] =   key256[4*i+1];
               w256[4*i+2]= key256[4*i+2];w256[4*i+3] =   key256[4*i+3];
         }
        byte* temp = new byte[4];
        for(i=Nk;i<4*(Nr+1);i++)
        {   temp[0]=w256[4*i-4];                          // The column before the current column
          temp[1]=w256[4*i-3];temp[2]=w256[4*i-2];
            temp[3]=w256[4*i-1];
           if(i%Nk==0)   // At every Nk and its multiples, special treatment is conducted for the column before the current column
            { temp=SubWord(RotWord(temp));
              // Shift fistly, then substitution to obain the constant from last round or
              temp[0]=(byte)((int)temp[0] ^ (int) AesRcon[4*(i/Nk)+0] );
              temp[1]=(byte)((int)temp[1] ^ (int) AesRcon[4*(i/Nk)+1] );
              temp[2]=(byte)((int)temp[2] ^ (int) AesRcon[4*(i/Nk)+2] );
              temp[3]=(byte)((int)temp[3] ^ (int) AesRcon[4*(i/Nk)+3] );
            }
          else if ( Nk > 6 && (i % Nk == 4) )
          {temp = SubWord(temp);}
           w256[4*i+0]=(byte)( (int) w256[4*(i-Nk)+0] ^ (int)temp[0] );
           w256[4*i+1]=(byte)( (int) w256[4*(i-Nk)+1] ^ (int)temp[1] );
           w256[4*i+2]=(byte)( (int) w256[4*(i-Nk)+2] ^ (int)temp[2] );
           w256[4*i+3]=(byte)( (int) w256[4*(i-Nk)+3] ^ (int)temp[3] );
        }
    }
```

4 Experimental Analysis

In order to verify the effectiveness of the iterative algorithm, computer simulation experiments need to be conducted, simulations carried out in the Matlab environment, in the form of simulation. The number of iterations is counted to observe whether it converge or not. The calculated results is shown as follows in Figure 1:

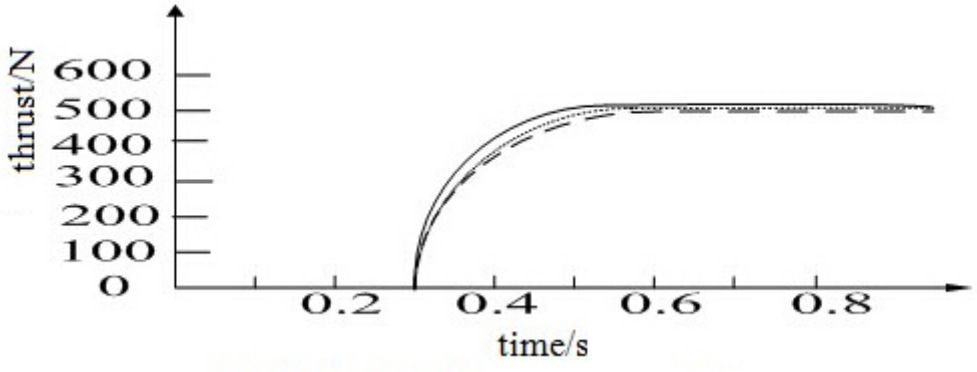

Figure 1 The convergence of the number of iterations

Figure 1 shows that after a certain time of iterating, the algorithm tends to converge, indicating that the proposed algorithm is valid, the introduction of self-learning algorithm can improve the efficiency and reduce the error rate, and provide ideas for studying further the issues related to the data processing.

5 Conclusions

An iterative optimization algorithm based on C language is designed and implemented in this paper, the algorithm is based on the traditional algorithm, and check constraint is added to ensure the correctness of the iterative process. The simulation software is regarded as the basis for computer simulation experiment, experimental results show that the algorithm converges.

References

[1] Yu Shaojuan, Qi Xiangdong, Wu Juhua. Theory and Applications of Iterative Learning Control. China Machine Press.2005

[2] Yang Junyou, Men Bo. Disturbance attenuation of permanent magnet linear synchronous motor based on iterative learning [J]. Journal of Shenyang University of Technology. 2010, 32(1):6-10.

[3] Ma Zhenzhen, Nan Yurong. Position Control for Permanent Magnet Linear Motor using Iterative Learning Control [J]. Small & special electrical machines, 2008(4): 49 -51

[4] Jin Xiaohua, Lin Jian. Analysis and Improvement for Thrust Fluctuation of Permanent Magnet Linear Synchronous Motor. Manufacturing information engineering of Chnia, 2007, 36(19)

[5] Tan K K,Zhao S.Adaptive force ripple suppression in iron-corepermanent magnet linear motors[C].Proceedings of the 2002 IEEE,Vancouver,Canada,2002:266-269