文章编号:1005－0523(2006)01－0113－04

基于 MATLAB 的全波傅氏算法仿真

何志勤,樊江涛

(华东交通大学 电气与电子工程学院,江西 南昌 330013)

摘要:运用 MATLAB 对传统全波傅氏算法和 2 种改进算法进行仿真,通过对 3 种算法频谱图的比较分析,说明 2 种改进傅氏算法能够有效滤掉故障电流中衰减的直流分量,从而获得更为精确的基波和谐波的幅值.

关 键 词:傅氏算法;衰减直流分量;MATLAB

中图分类号:TM774 **文献标识码**:A

1 引言

任何一种保护功能的实现都必须要有相应的算法来支持.我们对算法性能优劣的评价也取决于该算法能否在较短数据窗中从交流信号的若干采样值中获得基波分量或者某次谐波分量较为精确的估计值.傅氏算法是当今较常用的算法,该算法具有很强的滤除谐波分量的能力,但缺点是其本身不能滤除衰减的直流分量.为了克服信号中衰减的非周期分量的影响,国内外很多学者作了大量的分析研究工作,并提出了一些相应的保护算法.本文利用 MATLAB 对传统全波傅氏算法和 2 种改进后的傅氏算法进行仿真,并对仿真结果进行比较和探讨.

2 传统傅氏算法及其仿真

2.1 传统傅氏算法的推导

我们以电流为例,设定故障电流波形为以下形式:

$$i(t)=I_0e^{-\alpha t}+\sum_{k=1}^{n}I_k\sin(k\omega t+\varphi_k)$$

其中 k 为谐波次数,ωt 为基波的角频率.设 $I_{Rk}=I_k\sin(\varphi_k)$,$I_{ik}=I_k\cos(\varphi_k)$

则原式转化为:

$$i(t)=I_0e^{-\alpha t}+\sum_{k=1}^{n}[I_{Rk}\cos(k\omega t)+I_{1k}\sin(k\omega t)]$$

运用全波傅氏算法,有:

$$\begin{cases}a_k=\dfrac{2}{T}\int_0^T i(t)\cos(k\omega t)\mathrm{d}t\\ b_k=\dfrac{2}{T}\int_0^T i(t)\sin(k\omega t)\mathrm{d}t\end{cases}$$

经过 A/D(模拟量/数字量)变换采样量化后,连续量变为离散量,连续量求积变为离散量求和,从而有:

$$\begin{cases}a_k=\dfrac{2}{N}\sum_{n=1}^{N}i_n\cos(nk\dfrac{2\pi}{N})\text{式(1)}\\ b_k=\dfrac{2}{N}\sum_{n=1}^{N}i_n\sin(nk\dfrac{2\pi}{N})\text{式(2)}\end{cases}$$

上式中,N 为一个周期 T 内的采样点数;k 是谐波次数;n 为离散的采样点.

2.2 传统傅氏算法的仿真

在用 MATLAB 进行仿真模拟时,要注意 N 数目的选择,这将对仿真效果产生影响.仿真程序的流程图如图 1.

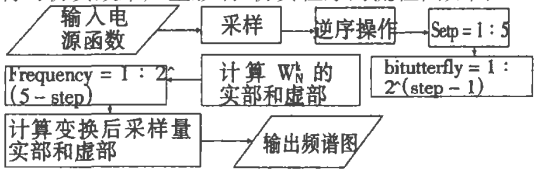

图 1 传统傅氏算法程序流程图

仿真程序的部分代码如下:

```
%FFT
global step %运算步数
global bitbutterfly %每个蝴蝶结中所包含的点数
```

收稿日期:2005－05－25

作者简介:何志勤(1982－),男,江西省九江市人,华东交通大学 03 级硕士研究生,研究方向为电力系统及其自动化.

```
global frequency %每步运算中蝴蝶结的个数
for step=1:5
for bitbutterfly=1:2^(step-1)
i = (2^(5-step)) * (bitbutterfly-1);RW = cos(2 * pi * i/32); % W 的实部
IW=(-1) * sin(2 * pi * i/32); % W 的虚部
for frequency=1:2^(5-step)
temp=(frequency-1) * (2^step)+bitbutterfly;
TR = dataR(temp);TI = dataI(temp);
dataR(temp)=dataR(temp)+RW * dataR(temp+2^(step-1))-IW * dataI(temp+2^(step-1));
dataI(temp)=dataI(temp)+RW * dataI(temp+2^(step-1))+IW * dataR(temp+2^(step-1));
temp1 = dataR(temp+2^(step-1));
dataR(temp+2^(step-1))=TR-(RW * dataR(temp+2^(step-1))-IW * dataI(temp+2^(step-1)));
dataI(temp+2^(step-1))=TI-(RW * dataI(temp+2^(step-1))+IW * temp1);
end
end
end
```

笔者选择 N 为 32 点，并选取两组故障电流进行仿真，从而比较非周期的衰减直流分量对运行结果的影响．最后观察的是每次谐波的幅值和真实值的差距，根据式(1)和式(2)，幅值即为：$|X(k)|=\sqrt{a_k^2+b_k^2}$．

3 两种改进后的傅氏算法及其仿真．

3.1 传统傅氏算法的误差来源

传统傅氏算法的误差主要来源于两方面：1)用离散值累加代替连续积分，其结果受到采样频率的影响．此外计算要用到全部 N 个采样值，因此，计算须在故障后第 N 个采样值出现时，才正确．在此之前，N 个采样值中有一部分吻故障前的数值，从而使结果不精确．2)传统傅氏算法无法滤掉衰减的直流分量．

3.2 滤除衰减直流分量的改进全波算法 1

所谓改进就是在全波傅氏变换提取出基波或各次谐波分量的基础上，减去直流衰减分量带来的误差．我们将实部和虚部的误差设为 δ_a 和 δ_b，则有：$\begin{cases} a_k=I_{Rk}+\delta_a \text{ 式(3)} \\ b_k=I_{ik}+\delta_b \text{ 式(4)} \end{cases}$；设 $\delta=i(0)-i(N)$，也就是说，在传统傅氏算法的数据窗 32 个点的基础上再加一个点，取第一个点减去第 $N+1$ 个点．通过计算，可得校正量：$\delta_a=2/(1+4\pi^2k^2K_2)I^*(0)$；$\delta_b=2k\pi\delta_a$；其中 $K=I^*(0)/\delta$．在交流采样算法中，$I^*(0)=\frac{1}{N}\sum_{n=0}^{N-1}i(n)$，即采样点数值之和与总采样点数之商．将计算出来的 δ_a 和 δ_b 回带入式(3)和式(4)，计算 a_k 和 b_k．仿真程序的流程图如图 2：

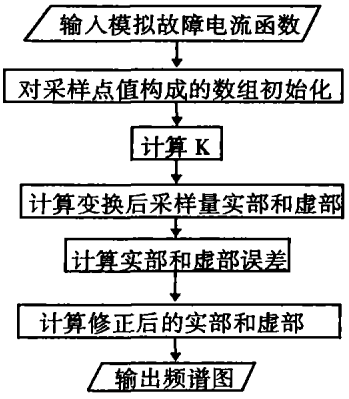

图 2 改进算法 1 程序流程图

仿真程序的部分代码如下：

```
p=3;sum=0;
for j=1:32 %计算 I(0)
sum=x(j)+sum;
end
y=sum/32;d=x(1+p)-x(33+p);
z=y/d; %计算 K
for k=1:32
dataR(k)=0;dataI(k)=0;
for n=1+p:32+p
datar(n)=x(n) * cos(2 * pi * n * k/32);datai(n)=x(n) * (-1) * sin(2 * pi * n * k/32);
datarr(n)=2 * y/(1+4 * pi * pi * k * k * (z^2)); %计算实部误差
dataii(n)=k * 2 * pi * z * datarr(k); %计算虚部误差
dataR(k)=dataR(k)+datar(n)-datarr(n);dataI(k)=dataI(k)+datai(n)-dataii(n);
end
end
```

程序中由 p 来控制起始采样点的位置，通过对 p 的调整，来测试其对运行结果的影响．结果说明 p=3 时的仿真效果比较好．

3.3 滤除衰减直流分量的改进全波算法 2

我们对输入信号进行等间隔采样．选取三个数据窗，时间间隔为一个采样周期．可以得到：$\begin{cases} a_k=I_k\cos\varphi_k+\delta_a \\ b_k=I_k\sin\varphi_k+\delta_b \end{cases}$；$\begin{cases} a_k'=I_k\cos(\varphi_k+k\omega\triangle t)+\delta_a' \\ b_k'=I_k\sin(\varphi_k+k\omega\triangle t)+\delta_b' \end{cases}$；$\begin{cases} a_k''=I_k\cos(\varphi_k+2k\omega\triangle t)+\delta_a'' \\ b_k''=I_k\sin(\varphi_k+2k\omega\triangle t)+\delta_b'' \end{cases}$；令：$A=a_k'-k_ca_k+k_sb_k$；$B=b_k'-k_cb_k+k_sa_k$；$C=a_k''-k_ca_k'+k_sb_k'$；$D=b_k''-k_cb_k'+k_sa_k'$；

进一步得出：$k_T=\frac{|C|+|D|}{|A|+|B|}$；$\delta_a=\frac{A(k_T-k_c)-Bk_s}{1+k_T^2-2k_ck_T}$；$\delta_b=\frac{B(k_T-k_c)-Ak_s}{1+k_T^2-2k_ck_T}$；

其中：$k_s=\sin(\frac{2\pi}{N}k)$；$k_c=\cos(\frac{2\pi}{N}k)$；将计算出来的 δ_a 和 δ_b

回带入式(3)和式(4)，计算 a_k 和 b_k．仿真程序的流程图如图 3．

仿真程序的部分代码如下：

for $n=(2+p):(33+p)$ % 取数据窗 3

datar3$(n)=x(n)*\cos(2*pi*n*k/32)$；datai3$(n)=x(n)*(-1)*\sin(2*pi*n*k/32)$；

data$R3(k)$=data$R3(k)$+datar3(n)；data$I3(k)$=data$I3(k)$+data$i3(n)$；

end

aa=data$R1(k)$−data$R2(k)$；% 计算 A

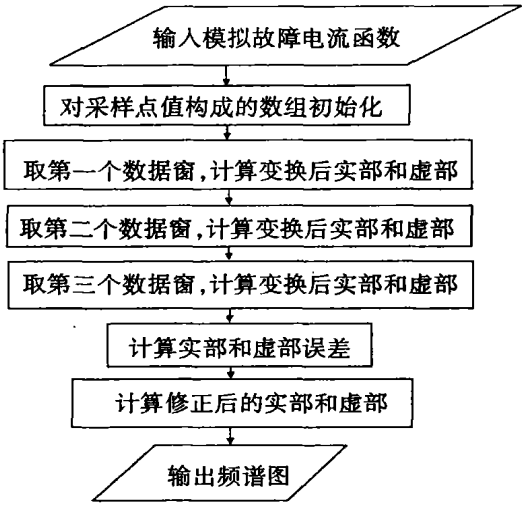

图 3　改进算法 2 程序流程图

bb=data$I1(k)$−data$I2(k)$；% 计算 B

cc=data$R2(k)$−data$R3(k)$；% 计算 C

dd=data$I2(k)$−data$I3(k)$；% 计算 D

$kt=abs(cc)/(abs(kc*aa+ks*bb))$；

$kc1=(bb*(kc*kt-1)+aa*ks*kt)/(1+kt\hat{}2-2*kt*kc)$；%虚部误差

$ks1=(aa*(kc*kt-1)-bb*ks*kt)/(1+kt\hat{}2-2*kt*kc)$；%实部误差

data$R(k)$=data$R1(k)+ks1$；data$I(k)$=data$I1(k)+kc1$；

end

笔者在编程时，对误差计算做了修改，结果更佳．要注意误差和初值的加减问题，在算法 2 的程序中误差是和初值相加，在算法 1 中是相减，否则结果误差会反而增大，此处不再做深入分析．

4　传统算法和改进算法的性能比较（N 统一取 32）

4.1　含有较大的衰减直流分量

根据程序，我们取：

$i(t)=30e^{-40t}+30\sin(100\pi t+\pi/3)+6\sin(200\pi t+\pi/4)+15\sin(300\pi t+\pi/6)+5\sin(400\pi t+\pi/3)+10\sin(500\pi t+\pi/4)$；

仿真结果如表 1．

表 1　含较大衰减值流分量仿真结果表

算法类型	基波		二次谐波		三次谐波	
	幅值	误差	幅值	误差	幅值	误差
传统傅氏算法（*FFT*）	33.8 215	12.74%	8.4 200	40.33%	16.7 708	11.81%
改进算法 1	30.4 495	1.50%	6.1 909	3.18%	14.4 796	−3.47%
改进算法 2	30.0 000	0.00%	6.0 000	0.00%	15.0 000	0.00%

计算后的仿真数值波形如图 4：

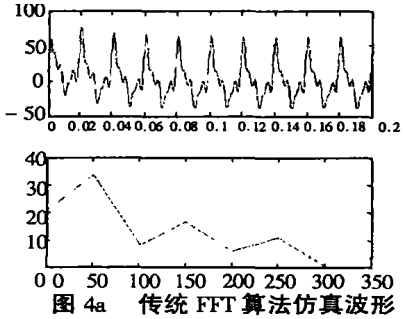

图 4a　传统 FFT 算法仿真波形

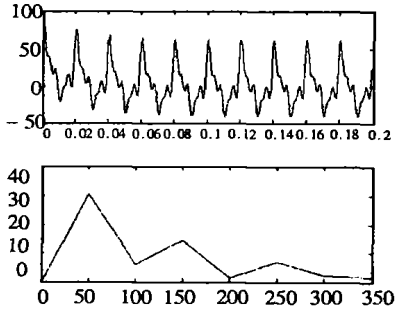

图 4b　改进算法 1 仿真波形

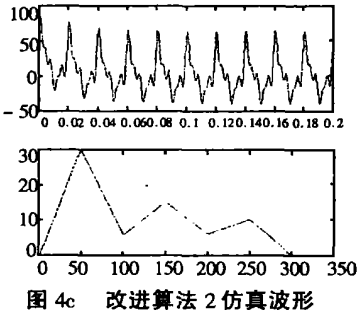

图 4c　改进算法 2 仿真波形

改进算法 2 仿真波形

4.2　含有较小的衰减直流分量

根据程序，我们取：

$i(t)=8e^{-20t}+30\sin(100\pi t+\pi/3)+6\sin(200\pi t+\pi/4)+15\sin(300\pi t+\pi/6)+5\sin(400\pi t+\pi/3)+10\sin(500\pi t+\pi/4)$；

仿真结果如下表 2：

表 2 含较小衰减值流分量仿真结果表

算法类型	基波		二次谐波		三次谐波	
	幅值	误差	幅值	误差	幅值	误差
传统傅氏算法(FFT)	30.5 413	1.8%	6.3 645	6.08%	15.2 796	1.86%
改进算法 1	30.0 170	0.06%	5.9 387	−1.02%	14.8 947	−0.70%
改进算法 2	30.0 000	0.00%	6.0 000	0.00%	15.0 000	0.00%

计算后的仿真数值波形如图 5.

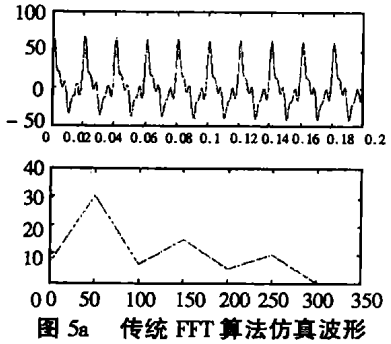

图 5a 传统 FFT 算法仿真波形

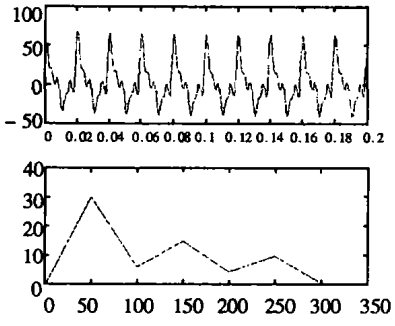

图 5b 改进算法 1 仿真波形

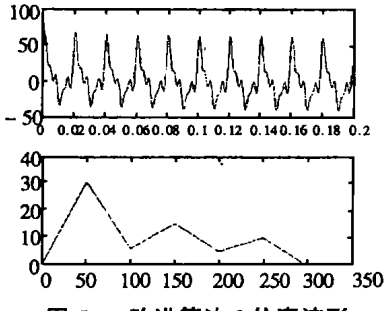

图 5c 改进算法 2 仿真波形

5 全波傅氏算法小结

全波富氏算法能滤除所有的整次谐波分量,且稳定性好,但其数据窗较长,所以其响应速度较慢.通过上面的仿真结果,我们可以得出以下结论:

1) 对照同一故障波形进行分析,可以看出:2 种改进算法所算结果都比较传统傅氏算法要精确.

2) 在故障波形谐波含量都相同的情况下,衰减直流分量的初始值越小,衰减时间常数越大时,衰减非周期分量对基波以及各次谐波的影响越小.

3) 笔者将采样点数 N 从 16 提高到 32 后,3 种算法误差均有一定程度的减小.如果进一步提高采样率到 64 点/周或 128 点/周时,对提高计算精度作用不大,而计算时间相应增加.因此,将采样点定为 32 点/周对减少误差有益.

虽然改进算法总体要比传统算法精确,但都还存在一个无法避免的计算误差,那就是所取数据窗内的正常数据和故障数据的分辨问题.为了全部使用故障后的采样值,往往取 $i \geqslant N/2$, i 表示从故障起始时刻开始第 i 个采样点.本文中是通过对改进算法中 p 的调整,模拟故障发生后,起始采样点的位置.理论上如果有充足的判据是可以的,但无疑增加了保护计算的时延和难度,而且跳过的半个周期采样值不予计算,这肯定会给故障时刻电流的计算带来一定的误差.因此,这个问题还有待进一步的研究.

参考文献:

[1] 刘建刚,孙同景. 基于 FFT 的傅立叶算法在微机继电保护中的应用[J]. 继电器. 2004,10:24−26.

[2] 李孟秋,王耀南,王辉. 基于全周波富氏算法滤除衰减直流分量新方法[J]. 湖南大学学报. 2001,1:59−63.

[3] 周大敏. 一种消除非周期分量对非递推富氏算法影响的精确方法[J]. 继电器. 1998,4:7−11.

Full-cycle Fourier Algorithm Emulation Based on MATLAB

HE Zhi-qin, FAN Jiang-tao

(School of Electrical and Electronic Eng., East China Jiaotong University, Nanchang 330013, China)

Abstract: This article make use of MATLAB to emulate traditional full-cycle Fourier algorithm and two kinds of improved Fourier algorithm, then analyze the spectrum picture of these algorithm. The result demonstrate that the improved Fourier algorithm can filter decaying DC component effectively. We can obtain the magnitude of fundamental wave and harmonic wave more accurate than traditional full-cycle Fourier algorithm.

Key words: Fourier algorithm; the decaying DC component; MATLAB (Matrix Laboratory)