实验二 交流系统模型

一、实验目的

1. 学习讲义，掌握交流系统原理及故障处理方式；
2. 结合上课内容复习重要知识点：故障元件及断路器设备建模；
3. 学习课程提供的Matlab实例程序；
4. 修改程序，完成交流系统故障实验的任务内容；
5. 完成实验报告。

二、实验原理

1.搭建如下交流系统模型：

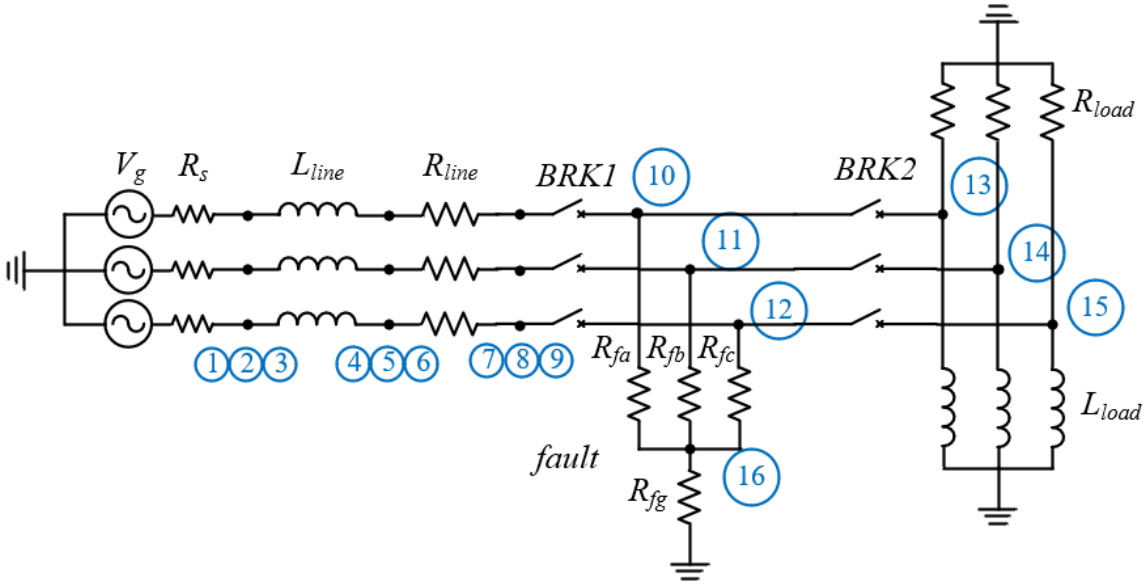

图1 交流系统拓扑模型

该系统中包含三相交流电源、三相线路（等效为RL支路）、两个三相断路器（BRK1、BRK2）、三相电阻负载、三相电感负载以及附加的故障模块（等效为电阻支路组合）。相关参数给出如下：

（1）电源参数：线电压有效值$V_g = 230V$，内阻$R_s = 0.001\Omega$，频率$f = 50Hz$；

（2）线路参数：线路电感$L_{line} = 0.05H$，线路电阻$R_{line} = 2\Omega$；

（3）负载参数：电感负载$L_{load} = 0.2H$，电阻负载$R_{load} = 50\Omega$；

（4）断路器参数、故障参数见后面说明。

2.在之前的EMTP框架基础上，增加了故障模块、断路器模块，其中：

（1）故障模块（$fault$）

故障模块等效模型如下图2所示。

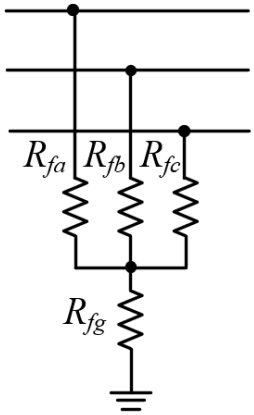

图2 故障模型

由图2，通过设置不同大小的$R_{fa}, R_{fb}, R_{fc}, R_{fg}$可用于模拟线路上不同的短路故障。比如A相接地故障，即可设置R_{fa}, R_{fg}为很小数值，R_{fb}, R_{fc}为较大数值（1e6）；若是三相接地故障，则设置$R_{fa}, R_{fb}, R_{fc}, R_{fg}$均为很小数值；其他故障可类似设置。当不添加故障模块时，可以设$R_{fa}, R_{fb}, R_{fc}, R_{fg}$均为默认较大数值（1e6），即等效为与电网脱离。在仿真中，因为故障等效为上述电阻支路，不包含储能元件，不用计算历史电流等，只需结合故障状态即时更新网络导纳矩阵即可。在已求解网络节点电压向量V_n的情况下，可由下式计算故障支路电流：

$$I_f = (V_n(n_1) - V_n(n_2))/R \tag{1}$$

其中，n_1, n_2为首、末端点，R为支路电阻，故障时取R_{on}值，无故障时取R_{off}值（见下）。

新增故障模块的参数格式如下：

n_1	n_2	$R_{on}-type$	$value$	R_{off}
首端点	末端点	故障支路类-型	故障发生时电阻值	无故障时默认电阻值

其中，故障支路类型包括$(1-R_{fa}, 2-R_{fb}, 3-R_{fc}, 4-R_{fg})$，$value$为对应设置的故障状态电阻值。本次实验中，考虑A相接地短路故障，此时参数设置如下：
$R_{fa}=0.1, R_{fb}=1e6, R_{fc}=1e6, R_{fg}=0.1$，$R_{off}=1e6$。

（2）断路器模块(BRK)

仿真中，考虑将断路器等效为三相电阻支路，其中闭合状态时电阻值取很小值R_{on}表示导通，断开状态时电阻值取较大值R_{off}表示开路。同样，仿真中不用计算其历史电流，只需结合断路器开关状态即时更新网络导纳矩阵即可。类似式（1），可计算断路器支路电流为

$$I_{BRK} = (V_n(n_1) - V_n(n_2))/R \tag{2}$$

其中，n_1, n_2为首、末端点，R为支路电阻，断路器闭合时取R_{on}值，断开时取R_{off}值。

需要注意的是，断路器只能在电流过零时才能断开，因为网络中存在电感元件，不允许电流突变。因此，需等待断路器电阻支路电流I_{BRK}过零时刻，才能更新网络导纳矩阵，反映断路器动作。然而，实际计算过程中，支路电流的过零点通常不在整数步长点上，此时需要利用线性插值法求出相应过零时刻。假定电流在$t_{n-1} \sim t_n$之间过零（可通过$I_{n-1} \cdot I_n < 0$判断），则可计算过零时刻为

$$t_0 = t_{n-1} + \frac{I_{n-1}}{I_{n-1} - I_n}\Delta t \tag{3}$$

其中，I_{n-1}, I_n符号相反，Δt为仿真步长。找出过零点t_0后，即可根据断路器的最新状态更新网络导纳矩阵。同时，也应利用线性插值公式计算出所有变量在t_0时刻的值，并修正当前时间为t_0，从该时刻开始进行后续仿真。本EMTP框架程序中，修正网络节点电压V_n、支路电流I_b向量即可，如下：

$$\begin{aligned} V_n^{n*} &= V_n^{n-1} + \frac{t_0 - t_{n-1}}{\Delta t}(V_n^n - V_n^{n-1}) \\ I_b^{n*} &= I_b^{n-1} + \frac{t_0 - t_{n-1}}{\Delta t}(I_b^n - I_b^{n-1}) \end{aligned} \tag{4}$$

其中，V_n^{n*}, I_b^{n*}即为修正的 t_0 时刻变量数值，需替代V_n^n, I_b^n参与后续仿真计算。

新增断路器模块的参数格式如下：

n_1	n_2	R_{on}	R_{off}	$init-state$
首端点	末端点	导通电阻	开路电阻	初始开关状态

其中，初始开关状态包括$(1-on, 0-off)$。本次实验中，参数统一设为：
$R_{on} = 0.1$，$R_{off} = 1e6$。

三、程序说明

0 文件列表

（1）data.m，输入网络参数、电源参数、故障参数以及断路器参数；

（2）loadcase.m，网络预处理过程，加载输入参数，获取节点数、支路数等中间数据，并初始化相应结构体内部数据，同时调用ybus.m 计算初始系统导纳矩阵；

（3）ybus.m，分别针对网络、电源、故障、断路器计算初始系统导纳矩阵；

（4）emtp_main.m，主体程序，调用上述函数完成初始化，并在主循环中完成EMTP各环节计算，包括调用update_fault.m、update_BRK.m 修改系统导纳矩阵，最后完成主循环，调用simuplot.m 绘制结果波形；

（5）update_fault.m，根据故障动作，修改系统导纳矩阵，详见注释说明；

（6）update_BRK.m，结合断路器动作，修改系统导纳矩阵，详见注释说明；

（7）simuplot.m，基于导出的结果记录数据，绘制结果波形；

（8）init_data.mat，提供的系统节点电压、支路电流初始值，自动加载，无需修改；

（9）simu_result.dat，程序导出的结果记录文件，每次运行结束自动生成，配合simuplot.m 可复现上次运行程序的仿真结果。

相比之前分压器实验EMTP框架，主要增加下列程序部分：

1.参数输入（data.m）

按上述定义格式输入故障参数、断路器参数。

```
% fault data
% n1 | n2 | Ron type(1-Rfa,2-Rfb,3-Rfc,4-Rfg) | value | Roff
%通过切换各故障支路电阻值来反映故障状态，详见任务书实验原理
gFault.mfault=[
10  16  1   0.1 1e6
11  16  2   1e6 1e6
12  16  3   1e6 1e6
16  0   4   0.1 1e6];

% BRK data
% n1 | n2 | Ron | Roff | init_state(1-close,0-open)
%通过切换断路器支路电阻值来反映断路器状态，详见任务书实验原理
gBRK.mBRK=[
7   10  0.1 1e6 1
8   11  0.1 1e6 1
9   12  0.1 1e6 1
10  13  0.1 1e6 1
11  14  0.1 1e6 1
12  15  0.1 1e6 1];
```

2.读取数据，相关结构体定义预设（loadcase.m）

```
%fault
n1=mfault(:,1);
n2=mfault(:,2);
idtmp=n2==0;
n2(idtmp)=nnode;  % conveter reference node number to max
idtmp=n1==0;
n1(idtmp)=nnode;% conveter reference node number to max
gFault.n1=n1;
gFault.n2=n2;
gFault.nfault=length(mfault(:,1));
gFault.Ron=mfault(:,4);   %故障发生时，各故障支路电阻值
gFault.Roff=mfault(:,5);  %故障清除时，各故障支路电阻值
gFault.Ib=zeros(gFault.nfault,1);  % 故障支路电流值

%BRK
gBRK.n1=mBRK(:,1);
gBRK.n2=mBRK(:,2);
gBRK.nBRK=length(mBRK(:,1));
gBRK.Ron=mBRK(:,3);   %断路器闭合时，相应支路电阻值
gBRK.Roff=mBRK(:,4);  %断路器开断时，相应支路电阻值
gBRK.state=mBRK(:,5);  %用于插值操作时，记录断路器当前状态
gBRK.flag=zeros(gBRK.nBRK,1);  %用于标记断路器切换状态，控制信号触发置1，切换完成置0
gBRK.Ib=zeros(gBRK.nBRK,1);  %断路器支路电流值，用于插值计算
```

3.根据初始状态，计算初始网络导纳矩阵（ybus.m）

```
% fault impact on Yn
%按初始无故障处理
gFault.Yeff=zeros(nfault,1);
n1=gFault.n1; % node1 index
n2=gFault.n2; % node2 index
for k=1:nfault
    R=gFault.Roff(k);
    y=1/R;
    gFault.Yeff(k)=y;
    Yn(n1(k),n1(k))=Yn(n1(k),n1(k))+y;
    Yn(n2(k),n2(k))=Yn(n2(k),n2(k))+y;
    Yn(n1(k),n2(k))=Yn(n1(k),n2(k))-y;
    Yn(n2(k),n1(k))=Yn(n2(k),n1(k))-y;
end

% BRK impact on Yn
%根据断路器初始状态处理
gBRK.Yeff=zeros(nBRK,1);
n1=gBRK.n1; % node1 index
n2=gBRK.n2; % node2 index
for k=1:nBRK
    if gBRK.state(k)
        R=gBRK.Ron(k);
    else
        R=gBRK.Roff(k);
    end
    y=1/R;
    gBRK.Yeff(k)=y;
    Yn(n1(k),n1(k))=Yn(n1(k),n1(k))+y;
    Yn(n2(k),n2(k))=Yn(n2(k),n2(k))+y;
    Yn(n1(k),n2(k))=Yn(n1(k),n2(k))-y;
    Yn(n2(k),n1(k))=Yn(n2(k),n1(k))-y;
end
```

4.仿真参数预设，设定故障时间、断路器动作时间（emtp_main.m）

```
%*********************************
%  simulation setting
%*********************************
gSim.dt=100e-6; % time step
gSim.Tmax=1; % maximum time

% fault
gFault.ctrl_fault=0; %0-no fault,1-add fault (故障控制信号，初始为0)
% Tfon=0.2;      %故障发生时间
Tfon=2*gSim.Tmax;
Tfoff=Tfon + 0.2;  %故障清除时间

%BRK
gBRK.enable_intplo=1; %使能断路器开断插值处理，即等待电流过零才断开
gBRK.ctrl_BRK1=1;  %1-closed 0-open (断路器控制信号，初始均为1)
gBRK.ctrl_BRK2=1;
Toff_BRK1=Tfon+0.1; %BRK1 断开时间，与故障发生绑定（即0.1s后断开）
Ton_BRK1=Tfoff+0.1; %BRK1 闭合时间，与故障清除绑定（即0.1s后闭合）
% Toff_BRK2=0.2; %BRK2 断开时间，切除负荷
Toff_BRK2=2*gSim.Tmax;
Ton_BRK2=Toff_BRK2 + 0.1; %BRK2 闭合时间，投入负荷
```

5.根据仿真时间修改故障、断路器控制信号，并调用相关函数修改网络导纳矩阵（emtp_main.m）

```
%**********************************
%  fault update Yn
%**********************************
gFault.last_ctrl_fault=gFault.ctrl_fault;  %保存上一步控制信号
if(t>= Tfon-dt && t<Tfoff-dt)  %根据预设的动作时间，修改控制信号
    gFault.ctrl_fault=1;
else
    gFault.ctrl_fault=0;
end
Yn=update_fault(Yn,Vn);  % 结合故障状态修改网络导纳矩阵

%**********************************
%  BRK update Yn
%**********************************
gBRK.last_ctrl_BRK1=gBRK.ctrl_BRK1; %保存上一步控制信号
gBRK.last_ctrl_BRK2=gBRK.ctrl_BRK2;
if(t>= Toff_BRK1-dt && t<Ton_BRK1-dt)  %根据预设的动作时间，修改控制信号
    gBRK.ctrl_BRK1=0;
else
    gBRK.ctrl_BRK1=1;
end
if(t>= Toff_BRK2-dt && t<Ton_BRK2-dt)  %根据预设的动作时间，修改控制信号
    gBRK.ctrl_BRK2=0;
else
    gBRK.ctrl_BRK2=1;
end
% 结合断路器状态修改网络导纳矩阵
[Yn,Vn,Ib,t]=update_BRK(Yn,Vn,Ib,VV(cnt-1,:)',CC(cnt-1,:)',t);
gSim.Yn=Yn;  %保存Yn，用于下一循环 EMTP 计算
```

6.根据故障状态修正网络导纳矩阵，并计算故障支路电流（update_fault.m）

```
function [Yn] = update_fault(Yn,Vn)
%根据故障动作，修改网络导纳矩阵
global gFault

n1=gFault.n1; % node1 index
n2=gFault.n2; % node2 index

%若故障控制信号发生改变，则修改Yn，否则跳过
if(gFault.last_ctrl_fault~=gFault.ctrl_fault)
    if(gFault.ctrl_fault) %根据当前控制信号，确定切换的电阻值
        R=gFault.Ron;
        R_last=gFault.Roff;
    else
        R=gFault.Roff;
        R_last=gFault.Ron;
    end
    gFault.Yeff=1./R; %同时更新支路电导，用于支路电流计算
    dy=1./R-1./R_last;    % 计算需要修正的导纳差值

    for k=1:gFault.nfault %修正Yn
        y=dy(k);
        Yn(n1(k),n1(k))=Yn(n1(k),n1(k))+y;
        Yn(n2(k),n2(k))=Yn(n2(k),n2(k))+y;
        Yn(n1(k),n2(k))=Yn(n1(k),n2(k))-y;
        Yn(n2(k),n1(k))=Yn(n2(k),n1(k))-y;
    end
end
gFault.Ib=gFault.Yeff.*(Vn(n1)-Vn(n2)); %计算故障支路电流
end
```

7.计算断路器支路电流，并应用线性插值方法寻找电流过零点，再根据断路器开关状态修正网络导纳矩阵，同时修正仿真时间、网络节点电压、支路电流向量（update_BRK.m）

```
function [Yn,Vn,Ib,t]=update_BRK(Yn,Vn,Ib,last_Vn,last_Ib,t)
% 结合断路器动作，修改网络导纳矩阵
global gSim gBRK
dt=gSim.dt;
n1=gBRK.n1; % node1 index
n2=gBRK.n2; % node2 index
gBRK.last_Ib=gBRK.Ib;
gBRK.Ib=gBRK.Yeff.*(Vn(n1)-Vn(n2));  % 更新断路器支路电流

if(gBRK.last_ctrl_BRK1~=gBRK.ctrl_BRK1) %根据 BRK1 动作修改当前状态及标志位
    id1=1:3;
    gBRK.state(id1,1)=1-gBRK.state(id1,1);
    gBRK.flag(id1,1)=[1;1;1];
end
if(gBRK.last_ctrl_BRK2~=gBRK.ctrl_BRK2) %根据 BRK2 动作修改当前状态及标志位
    id2=4:6;
    gBRK.state(id2,1)=1-gBRK.state(id2,1);
    gBRK.flag(id2,1)=[1;1;1];
end

if(any(gBRK.flag)) %若标志位不全为0，即表示还有断路器的动作切换需要完成，否则跳过
    id_tmp=(find(gBRK.flag==1))';  % 找出需要处理的断路器开关支路
    for k=id_tmp
        if(gBRK.enable_intplo && gBRK.state(k)==0) %已使能插值且当前开关需开断
            if(gBRK.last_Ib(k)*gBRK.Ib(k)<0) %插值判断，寻找支路电流过零点
                t0=t-dt+gBRK.last_Ib(k)/(gBRK.last_Ib(k)-gBRK.Ib(k))*dt;
                Vn=last_Vn + (t0-t+dt)/dt*(Vn-last_Vn);
                Ib=last_Ib + (t0-t+dt)/dt*(Ib-last_Ib);
                t=t0;           % 同步插值修正 Vn Ib t
                gBRK.Ib(k)=0;   % 插值后新的支路电流

                R=gBRK.Roff(k);    % 断路器开断
                R_last=gBRK.Ron(k);
                gBRK.Yeff(k)=1/R; % 更新支路导纳
                y=1/R-1/R_last;  % 计算修正量并修改Yn
                Yn(n1(k),n1(k))=Yn(n1(k),n1(k))+y;
                Yn(n2(k),n2(k))=Yn(n2(k),n2(k))+y;
                Yn(n1(k),n2(k))=Yn(n1(k),n2(k))-y;
                Yn(n2(k),n1(k))=Yn(n2(k),n1(k))-y;

                gBRK.flag(k)=0;   %完成该支路的切换处理，重置标志位
            end
        else   %未使能插值处理或当前开关需闭合
                if(gBRK.state(k))     % 根据当前开关状态确定切换后电阻值
                    R=gBRK.Ron(k);
                    R_last=gBRK.Roff(k);
                else
                    R=gBRK.Roff(k);
                    R_last=gBRK.Ron(k);
                end
                gBRK.Yeff(k)=1/R; % 更新支路导纳
```

```
                y=1/R-1/R_last;    % 计算修正量并修改Yn
                Yn(n1(k),n1(k))=Yn(n1(k),n1(k))+y;
                Yn(n2(k),n2(k))=Yn(n2(k),n2(k))+y;
                Yn(n1(k),n2(k))=Yn(n1(k),n2(k))-y;
                Yn(n2(k),n1(k))=Yn(n2(k),n1(k))-y;

                gBRK.flag(k)=0;    %完成该支路的切换处理，重置标志位
            end
        end
    end

    end
```

四、实验内容

仿真初始时刻，默认图1系统中断路器BRK1、BRK2均闭合，未发生故障。在此基础上，可按要求完成下述实验。**注意：** 每一个实验的基本流程为：检查输入数据与系统初始状态 → 根据要求修改相应参数配置 → 运行程序 → 记录结果波形并分析结果，最终完成实验报告。

1.稳态运行实验

将故障发生时间、BRK2切负荷时间均设为超出仿真总时间，即

```
Tfon=2*gSim.Tmax;
Toff_BRK2=2*gSim.Tmax;
```

此时，系统稳态运行，记录节点10-12电压波形、线路电感L_{line}上电流波形。

示例：

给出系统稳态运行的相关波形（也即所提供的simu_result.dat记录的仿真结果）。

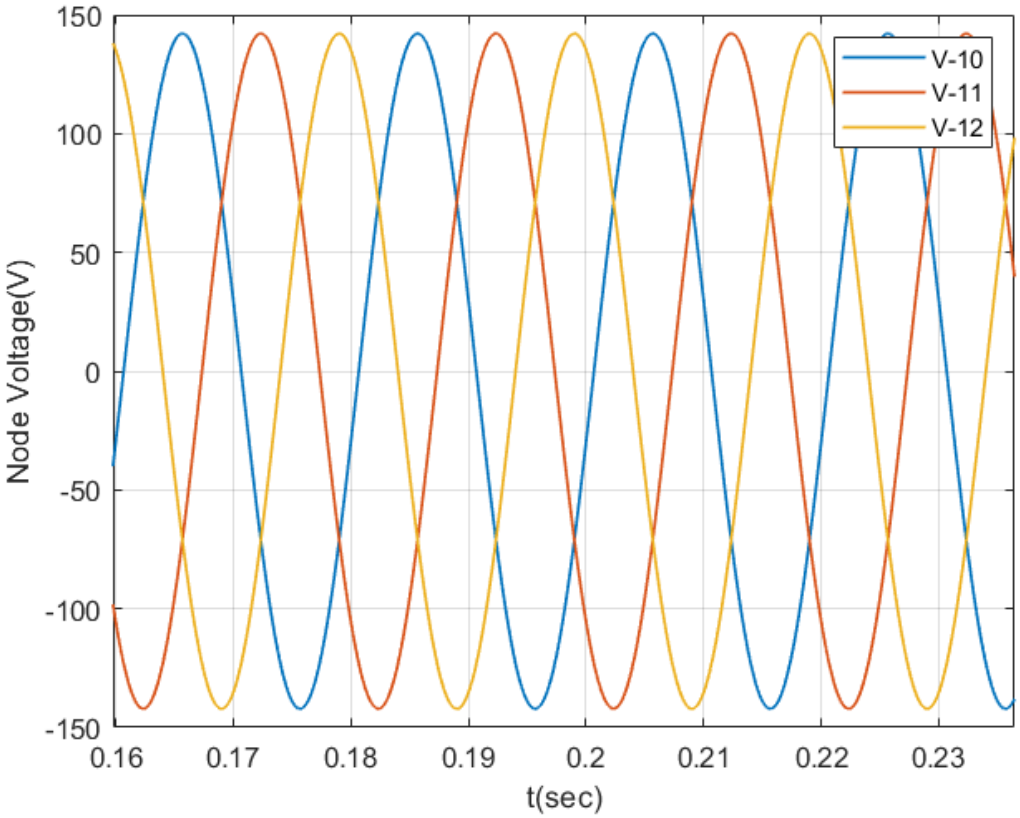

图3 稳态运行节点电压波形

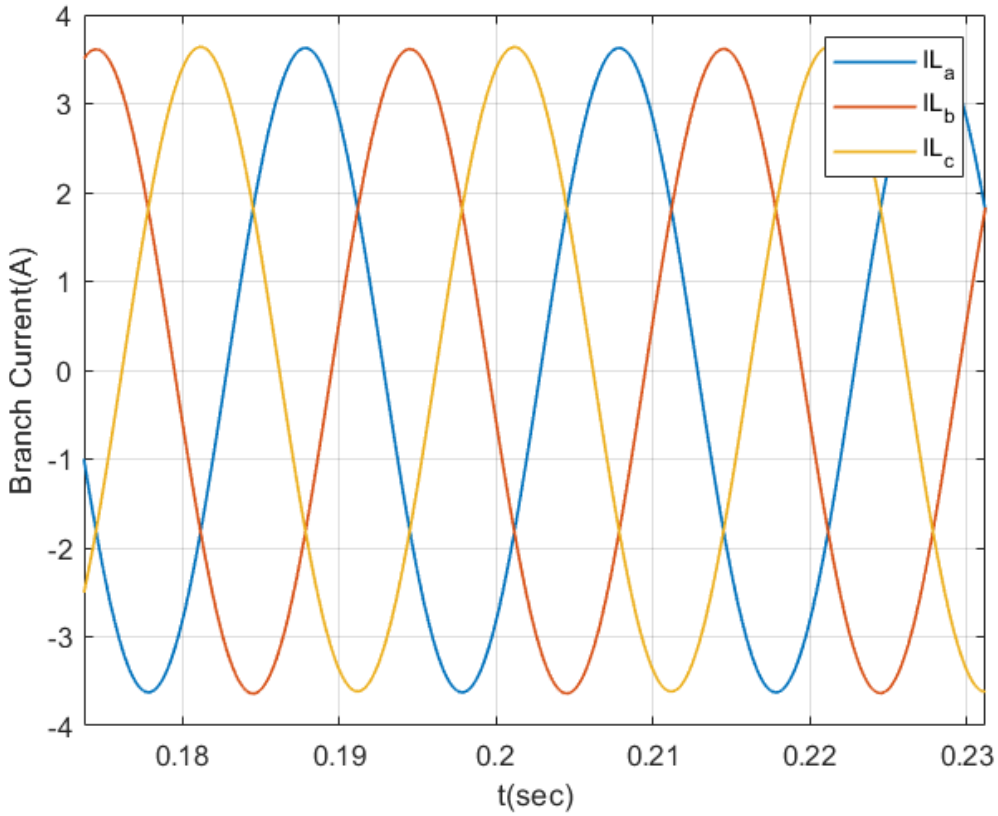

图4 稳态运行线路电流波形

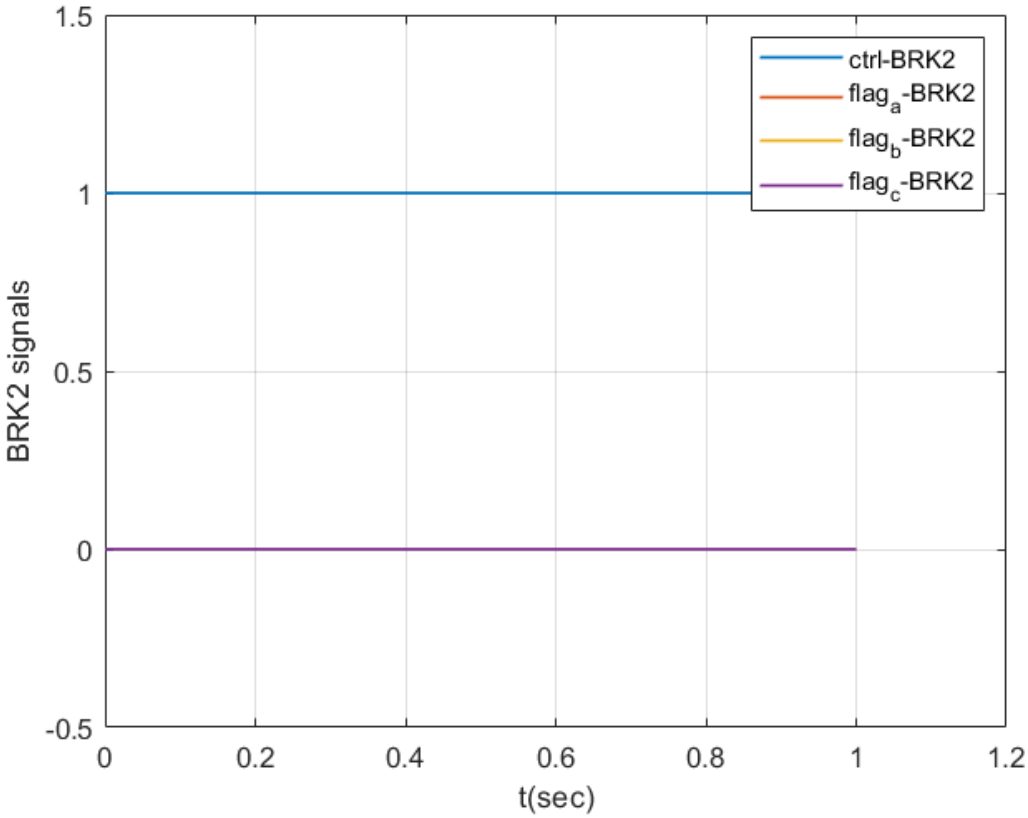

图5 稳态运行BRK2信号波形

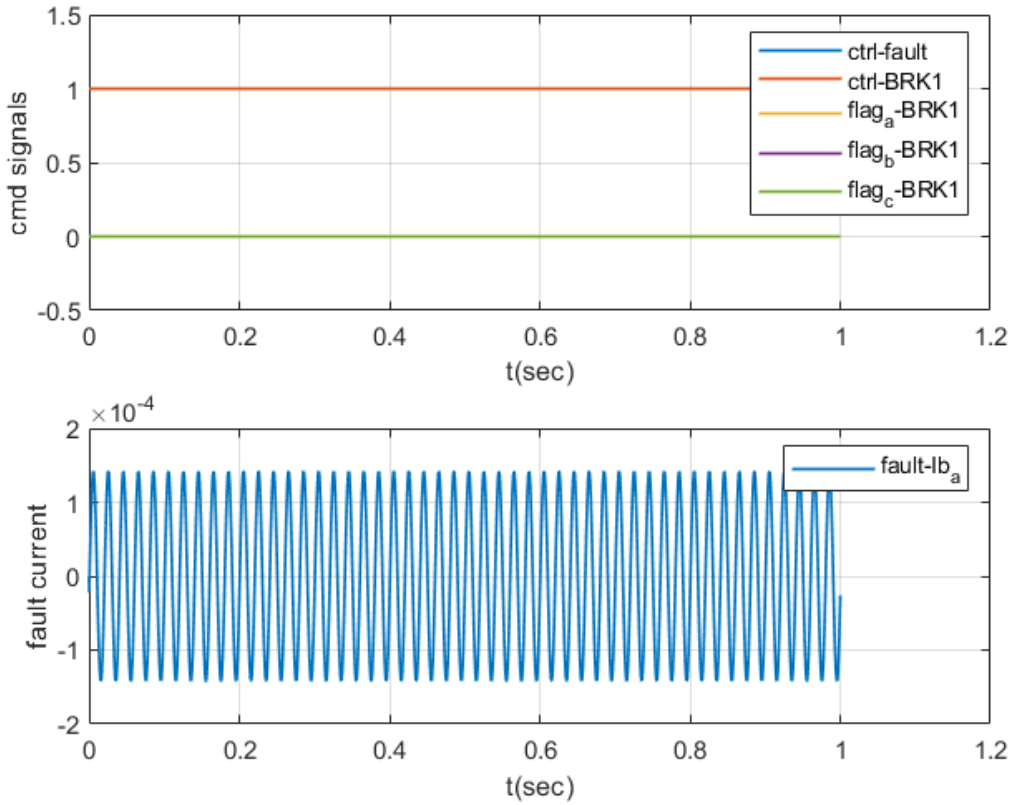

图6 稳态运行BRK1及故障信号波形、故障电流波形

图3、图4即为稳定运行的电压、电流波形。而此时未添加故障、未切除负荷、且各断路器均闭合，所以图5、图6中相关信号波形均为初始值，未发生改变，也没有故障电流。那么，结合下列实验内

容要求，请记录相应波形，并简要分析。

2.投/切负荷实验（一）

将故障发生时间设为超出仿真总时间，BRK2切负荷时间设为0.2s，且不使能断路器开断插值，即

```
Tfon=2*gSim.Tmax;
Toff_BRK2=0.2;
gBRK.enable_intplo=0;
```

此时，系统将在0.2s断开BRK2切负荷，而0.1s后再投入负荷。记录节点10-12电压波形、线路电感L_{line}上电流波形、断路器BRK2相关控制、标记信号波形。

3.投/切负荷实验（二）

重复上一实验，但使能断路器开断插值，即

```
Tfon=2*gSim.Tmax;
Toff_BRK2=0.2;
gBRK.enable_intplo=1;
```

记录节点10-12电压波形、线路电感L_{line}上电流波形、断路器BRK2相关控制、标记信号波形，并与上一实验结果进行对比，简要分析波形结果。

4.故障处理实验（一）

将BRK2切负荷时间设为超出仿真总时间，故障发生时间设为0.2s，使能断路器开断插值，即

```
Tfon=0.2;
Toff_BRK2=2*gSim.Tmax;
gBRK.enable_intplo=1;
```

此时，系统将在0.2s发生A相接地短路故障，而0.1s后BRK1将开始动作切除故障；0.4s时故障消失，再过0.1s后BRK1将开始动作合闸。记录节点10-12电压波形、线路电感L_{line}上电流波形、断路器BRK1相关控制、标记信号波形、故障控制信号波形及故障电流波形。结合波形结果，阐述故障动作及断路器动作过程，分析故障影响。

5.故障处理实验（二）

修改输入的故障参数矩阵，使故障类型变为三相短路故障。其他设置不变，重复上一实验。记录节点10-12电压波形、线路电感L_{line}上电流波形、断路器BRK1相关控制、标记信号波形、故障控制信号波形及故障电流波形。结合波形结果，阐述故障动作及断路器动作过程，分析故障影响。

*6.故障处理实验（三）（**拓展实验，选做**）

所给程序中，已将断路器BRK1的动作与故障动作绑定，即

```
Toff_BRK1=Tfon+0.1; %BRK1 断开时间，与故障发生绑定（即0.1s后断开）
Ton_BRK1=Tfoff+0.1; %BRK1 闭合时间，与故障清除绑定（即0.1s后闭合）
```

也即默认断路器BRK1能即时识别故障，并在故障动作0.1s 后相应动作。然而在实际交流系统中，当线路发生故障，往往需要保护装置来识别故障，进而触发相应断路器跳闸。那么，请设计一种故障识别方法，从而能根据程序中故障相应动作进行判断（不能直接根据 ctrl_fault 判断），进而控制断路器BRK1的跳闸及重合闸动作。此时，修改代码如下并补充完整程序，重新完成故障处理实验（一），并简要分析所设计方法的效果。

```
Toff_BRK1=2*gSim.Tmax;; %解除绑定，需在主循环中补充相应的ctrl_BRK1控制逻辑
Ton_BRK1=2*gSim.Tmax;; %解除绑定，需在主循环中补充相应的ctrl_BRK1控制逻辑
```

提示： 现给出两种思路供参考：

（1）监测故障支路的故障电流（或线路电流），一旦某时刻越过所设定的电流阈值，即可判断发生故障，进而控制BRK1动作。需设置合适的阈值，避开正常的负荷电流。事实上，该方法即为针对本实验简化的电流保护方法。

（2）采集电源处节点电压u及线路电流i，按下式计算其有效值U_{rms}, I_{rms}：

$$U_{rms} = \sqrt{\frac{1}{T}\sum_{i=1}^{n}(u_i^2\Delta t)} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}u_i^2}$$

$$I_{rms} = \sqrt{\frac{1}{T}\sum_{i=1}^{n}(i_i^2\Delta t)} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}i_i^2}$$

其中，u_i, i_i为各仿真步下的电压、电流值，$T = n\Delta t$为采样周期，一般为工频周期的整数倍，Δt为仿真步长。那么可计算网络等效阻抗为

$$|Z| = \frac{U_{rms}}{I_{rms}}$$

可以此作为衡量指标，设定相应阈值，判断故障是否发生。事实上，该方法即为针对本实验简化的距离保护方法。