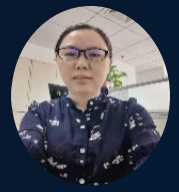
国家电网
STATE GRID

中国电力科学研究院
CHINA ELECTRIC POWER RESEARCH INSTITUTE

基于模型的风电并网性能在线评价系统设计

李春彦
中国电力科学研究院



MATLAB EXPO 2021

基于模型的风电并网性能在线评价系统设计

- 风电并网性能在线评价系统
- 基于模型的在线评价算法设计及C代码自动生成
- Linux环境下的代码集成
- 总结

中国电力科学研究院新能源中心

- 公司新能源发电**规划管理及并网运行**的技术支撑单位；
- **国内最早**从事新能源发电研究与咨询工作的机构之一。

1994

我国新能源发电刚刚起步，中国电科院成立新能源发电研究室。

2006

《可再生能源法》实施，中国电科院成立新能源研究所。

2009

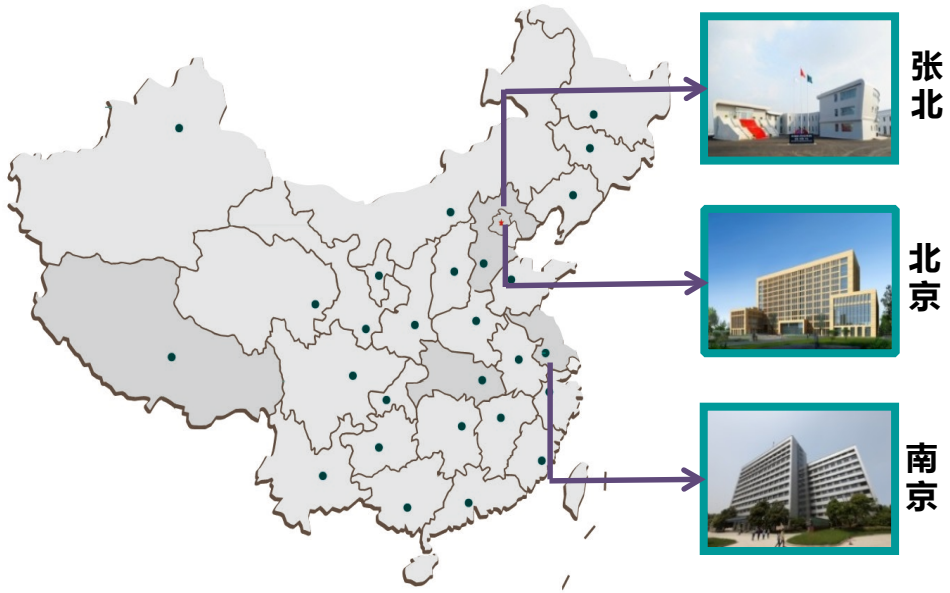
国网电科院成立清洁能源发电研究所，在公司大力支持下，开始建设张北试验基地。

2012

公司直属科研产业重组整合，中国电科院与国网电科院相关机构重组为新的新能源研究所。

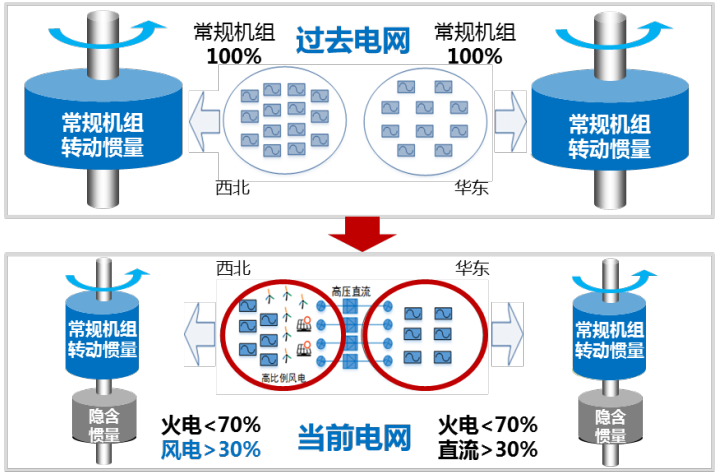
2016

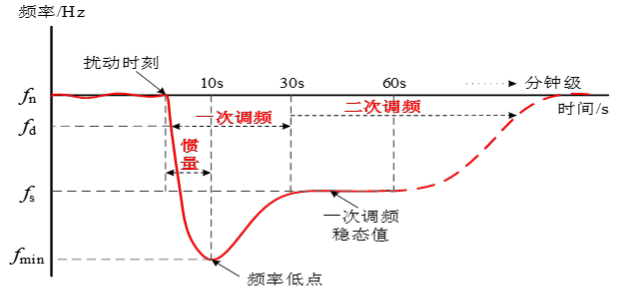
为加强新能源领域研究工作，公司党组批示以新能源研究所为基础成立新能源研究中心。

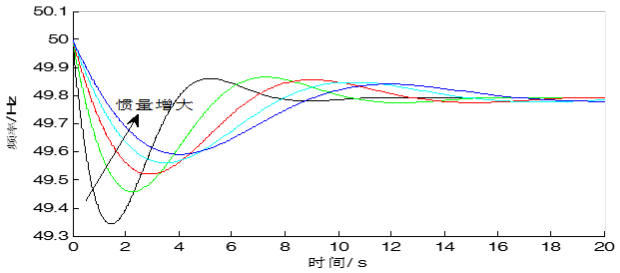


风电并网性能在线评价系统-项目背景

- 随着风电产业的快速发展，“三北”地区的高比例风电系统形态已基本形成，风电并网特性不足，高比例风电系统安全稳定水平持续下降；
- GB 38755-2019 电力系统安全稳定导则，对风电等新能源发电的惯量响应、一次调频、快速调压等电网友好型主动支撑性能进行了规范；
- 风电惯量响应、一次调频、无功调压等网源友好主动支撑性能的开发利用，亟需在线评估风电的实时惯量水平、有功/无功容量等直接影响频率/电压稳定的水平，以辅助调度决策。

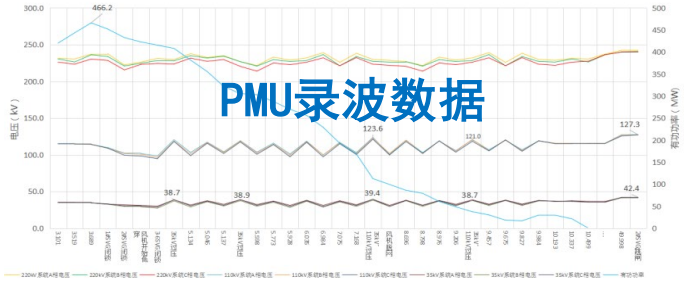


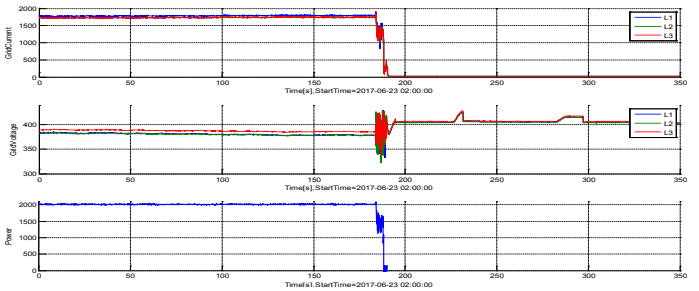




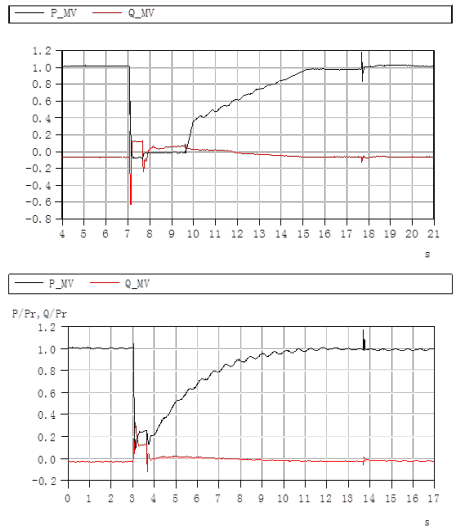
风电并网性能在线评价系统-项目背景

- 风机的并网运行产生大量的数据，尚未得到有效收集和全面利用，风电场运行监测存在“站级强，单机弱”的问题；
- 风电并网事故分析主要依靠系统同步相量测量装置（PMU），缺乏机端（690V）故障录波数据，电网末梢的感知功能尚待挖掘，急需建立风机级的PMU系统，以便进行大规模风电并网事故分析；
- 由于产品一致性的问题，风电的实际并网运行特性与试验差异较大，故障穿越、电能质量、功率控制、电网适应性等并网性能指标难以准确表征和管控。



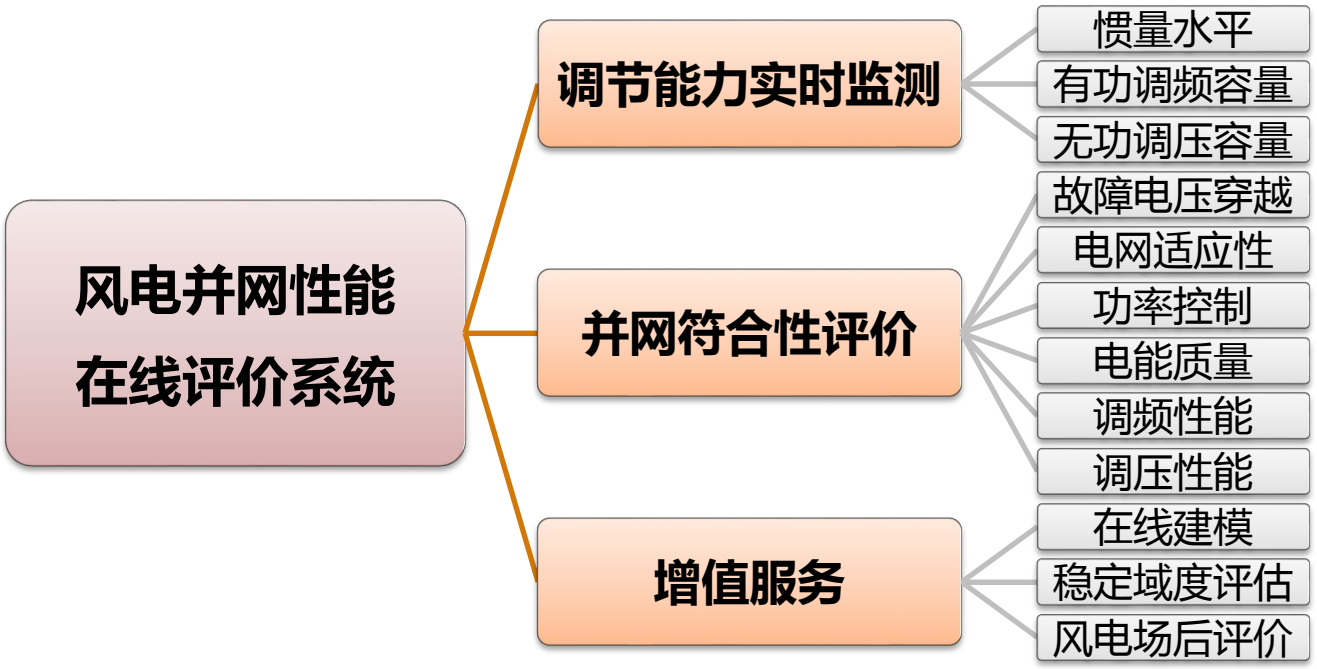


风电场次/超同步振荡数据分析



风电并网性能在线评价系统

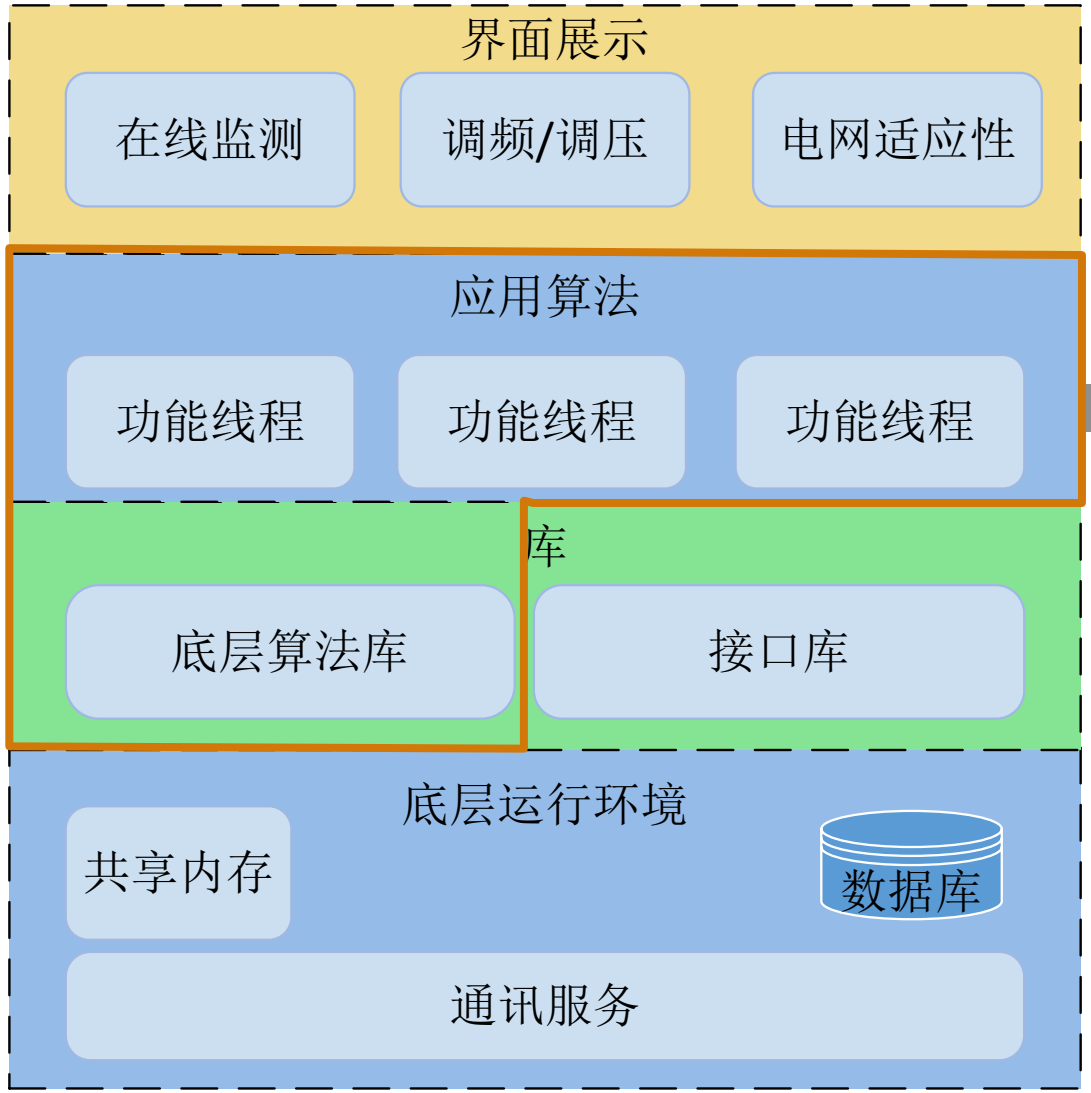
- 风电并网性能在线评价系统按照“全面采集、统一接口、分类利用”的原则，全面感知风电机组和风电场的运行状态，对风场的惯量水平、频率和电压调节能力进行实时监测，对电网适应性、功率控制、电能质量、调频调压性能等并网符合性进行在线评价。

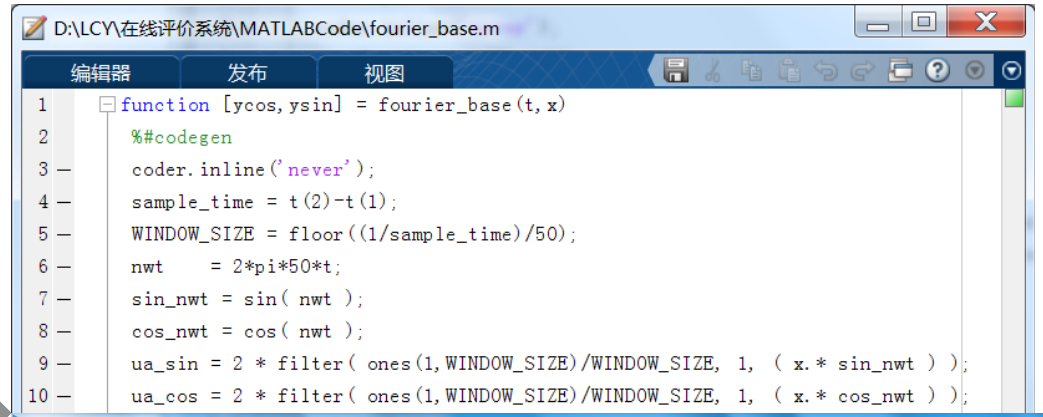


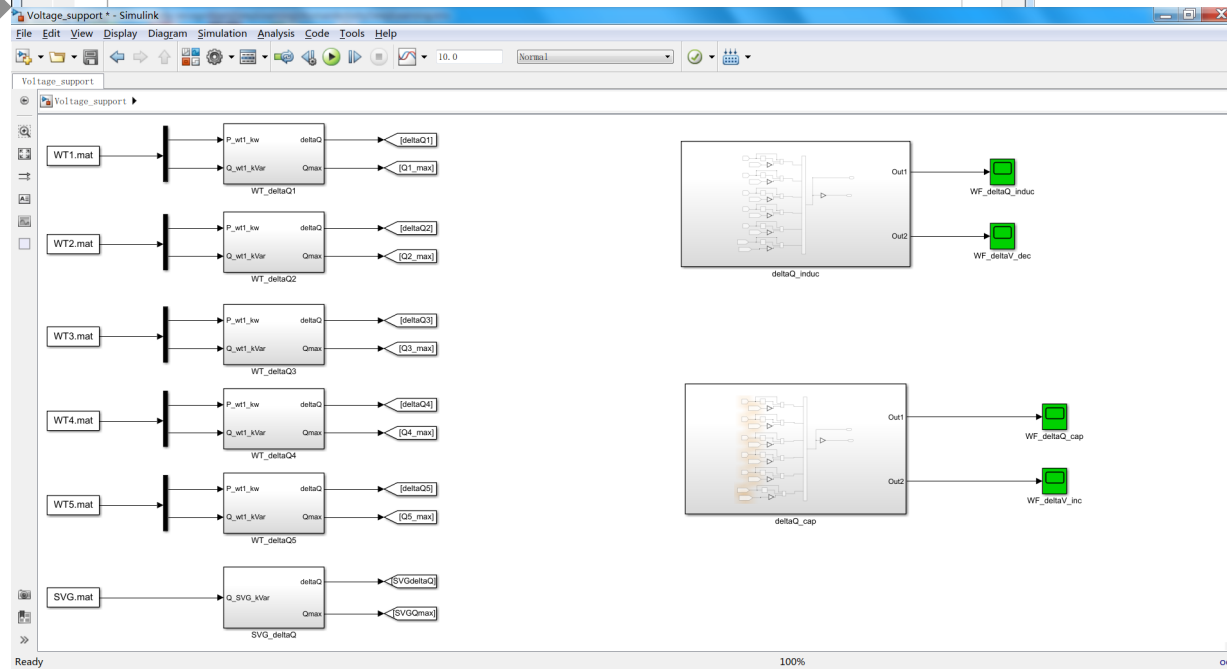
基于模型的风电并网性能在线评价系统设计

- 风电并网性能在线评价系统
- 基于模型的在线评价算法设计及C代码自动生成
- Linux环境下的代码集成
- 总结

基于模型的风电并网性能在线评价算法设计







风电场电能质量闪变计算模型设计

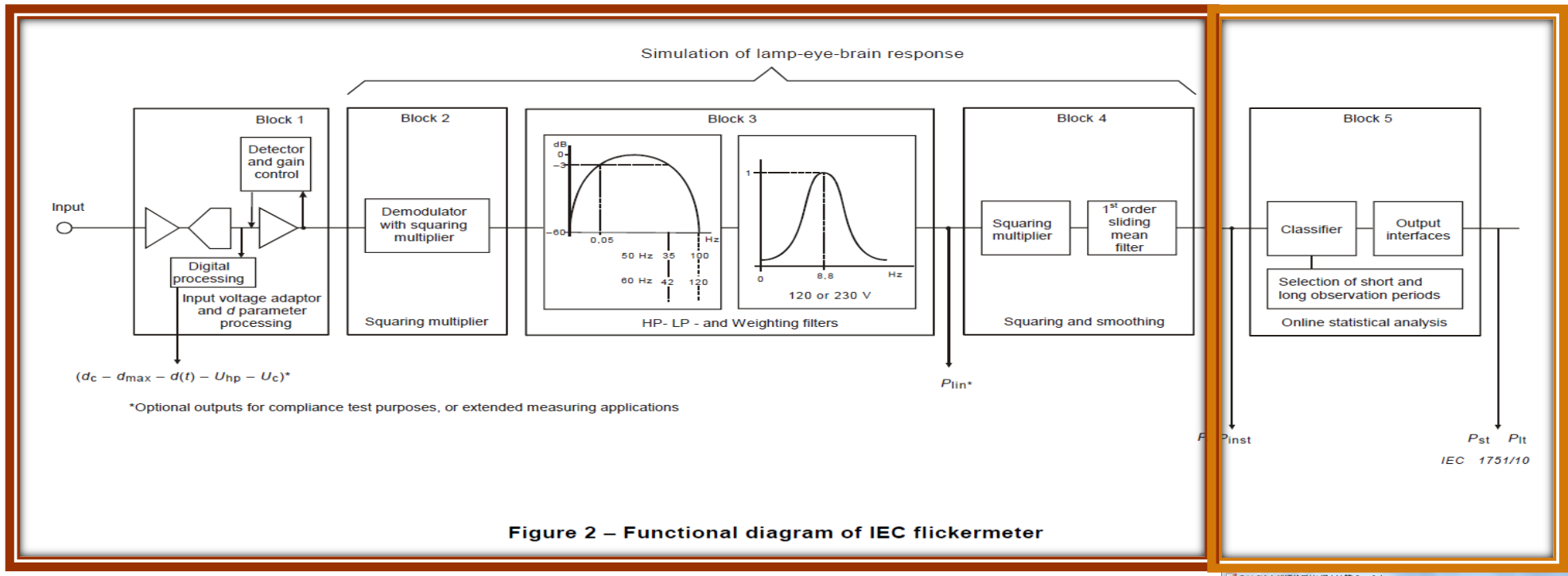
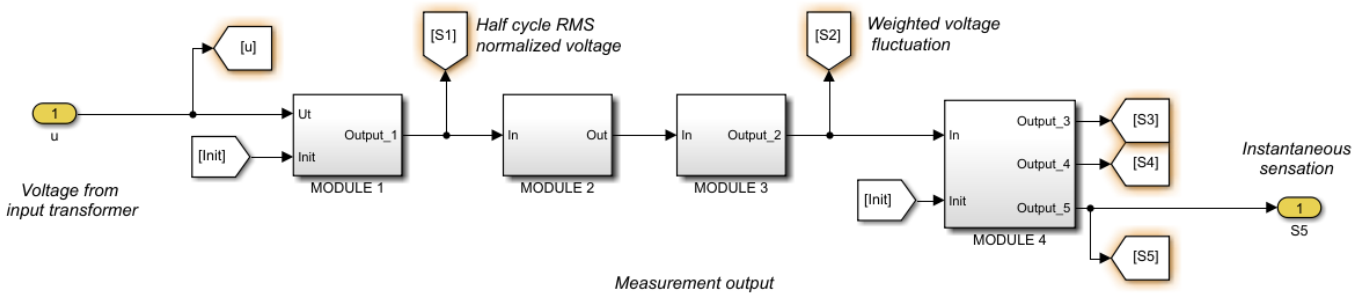
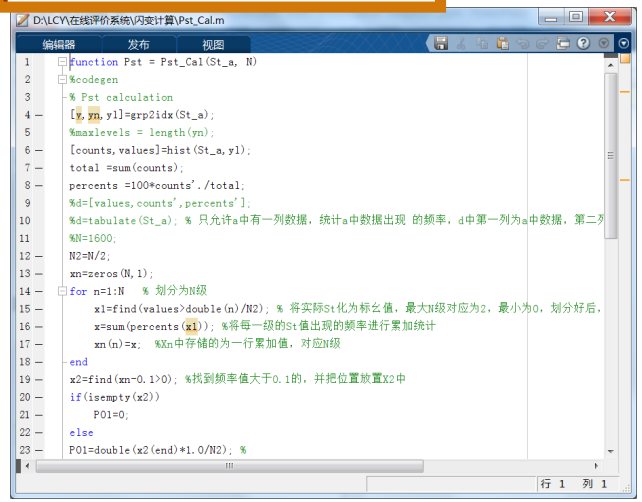


Figure 2 – Functional diagram of IEC flickermeter



SPS Digital Flickermeter 模块



风电场电能质量闪变计算及代码生成

D:\LCY\在线评价系统\闪变计算\Pst_Cal.m

编辑器 发布 视图

```
function Pst = Pst_Cal(St_a, N)
%codegen
% Pst calculation
[y,yn,yl]=grp2idx(St_a);
%maxlevels = length(yn);
[counts,values]=hist(St_a,yl);
total =sum(counts);
percents =100*counts'./total;
%d=[values,counts',percents'];
%d=tabulate(St_a); % 只允许a中有一列数据，统计a中数据出现 的频率，d中第一列为a中数据，第二列
%N=1600;
N2=N/2;
xn=zeros(N,1);
for n=1:N   % 划分为N级
    x1=find(values>double(n)/N2); % 将实际St化为标幺值，最大N级对应为2，最小为0，划分好后，
    x=sum(percents(x1)); %将每一级的St值出现的频率进行累加统计
    xn(n)=x;  %Xn中存储的为一行累加值，对应N级
end
x2=find(xn-0.1>0); %找到频率值大于0.1的，并把位置放置X2中
if(isempty(x2))
    P01=0;
else
P01=double(x2(end)*1.0/N2); %
```

行 1 列 1

M脚本

D:\LCY\在线评价系统\闪变计算\Flicker_Codegen_test_17a_110kv_6.4k_genCode\Flick...

编辑器 视图

```
/* Model step function */
void Flicker_Codegen_test_step(real_T *u, real_T *s5)
{
  int32_T denIdx;
  int32_T j;
  real_T rtb_fonction_j;
  real_T rtb_yk_g;
  real_T rtb_yk_a;
  real_T rtb_DiscreteFIRFilter;
  boolean_T rtb_RelationalOperator;
  real_T rtb_Switch;
  real_T MathFunction;
  real_T DiscreteTransferFcn_tmp_l;
  real_T DiscreteTransferFcn_tmp;
  real_T tmp;
  real_T u0;

  /* Outputs for Enabled SubSystem: '<S26>/POSITIVE Edge' incorporates:
   *  EnablePort: '<S28>/Enable'
   */
  /* RelationalOperator: '<S28>/Relational Operator1' incorporates:
   *  Constant: '<S2>/C2'
   *  DataTypeConversion: '<S25>/Data Type Conversion2'
   *  UnitDelay: '<S26>/Unit Delay'
```

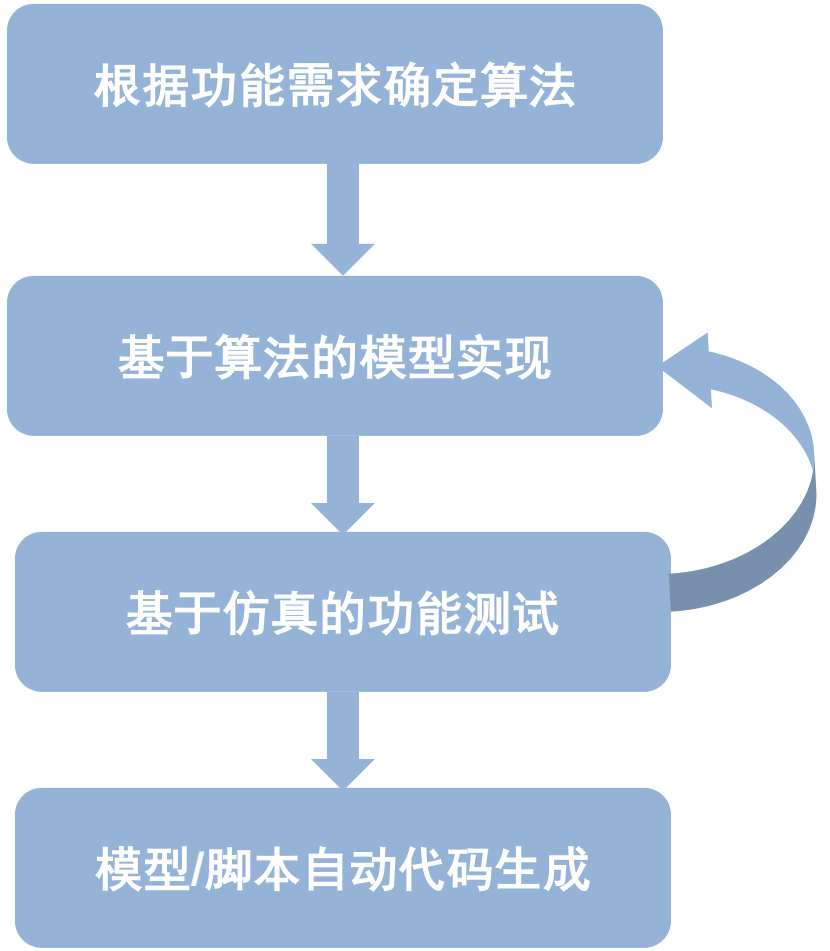
C/CPP 源或头文件 行 1 列 1

C代码

- 分别对SPS Digital Flickermeter和MATLAB脚本进行C代码生成。

基于模型的风电并网性能在线评价系统算法设计及代码生成

- 根据功能需求进行软件架构设计和功能模块划分；
- 大部分的算法只需直接使用SPS工具箱内的库模块（部分模块需要微小改动以支持C代码生成），极大地加速了模型实现；
- 软件功能测试占用了系统开发的大部分时间；
- 绝大部分（约90%）的库函数和功能函数均通过自动代码生成来实现。

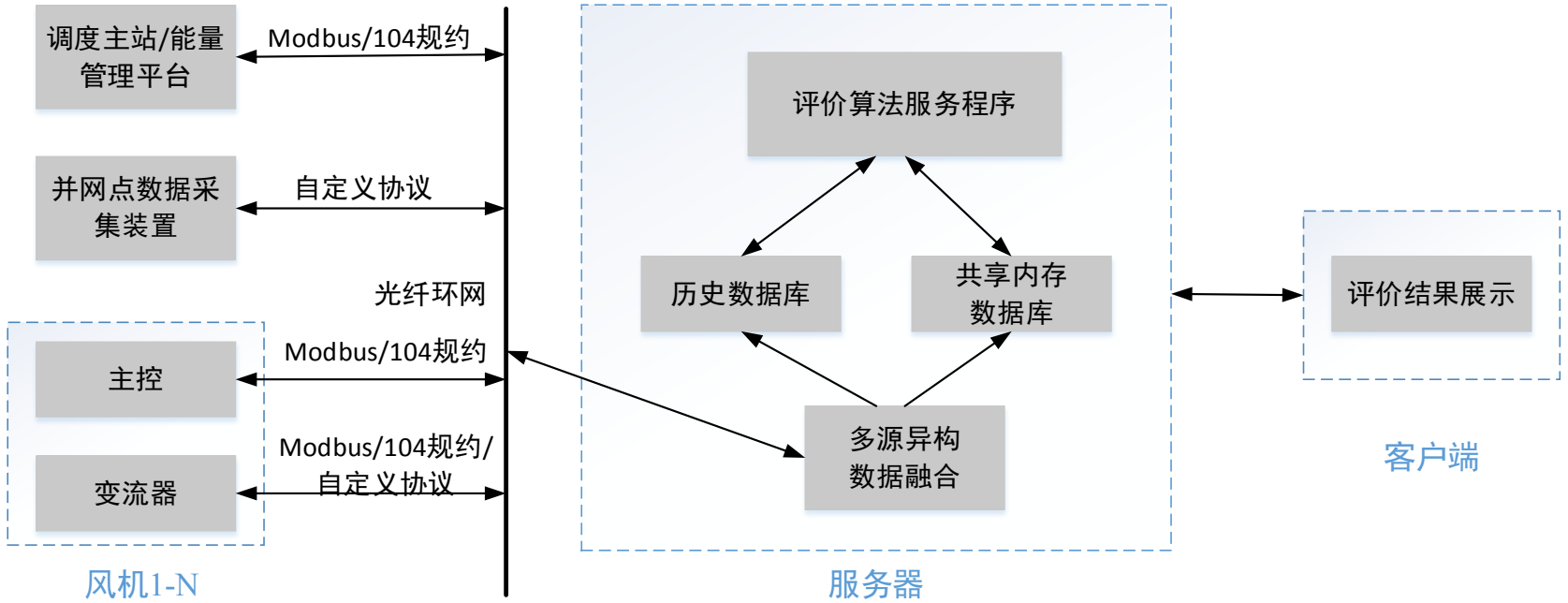


基于模型的风电并网性能在线评价系统设计

- 风电并网性能在线评价系统
- 基于模型的在线评价算法设计及C代码自动生成
- Linux环境下的代码集成
- 总结

Linux环境下代码集成

- 代码运行环境：Linux CentOS 操作系统；
- 采用手写框架代码并集成各功能函数的方式来实现评价算法服务程序。



风电并网性能在线评价系统结构示意图

Linux环境下代码集成

- 评价算法服务进程包含3个线程，评价算法按照计算周期要求分布于不同线程；
- 进程框架和各线程的数据读写接口采用手写代码，集成算法功能函数。

Flicker_Codegen_test_17a.c | readCSV.h | HWSqlProc.h | hist_codegen.c | myStruc

OnLineEval1124_aliTest | (全局范围)

```
2036    //HWSqlMemDBSimDataEnable(1);
2037    time(&now);
2038    timenow = localtime(&now);
2039    std::cout << "Online Evaluation Start at" << asctime(timenow) << endl;
2040
2041  #ifdef WIN32
2042    //初始化数据库
2043    if (HWSqlInit((int8*)"112.95.153.120") < 0)
2044  #else
2045    if (HWSqlInit((int8*)"127.0.0.1") < 0)
2046  #endif
2047    {
2048        cout << "HWSqlInit error!" << endl;
2049  #ifdef WIN32
2050        system("pause");
2051  #endif
2052        return 0;
2053    }
2054
2055    HWSqlCreateThread((void*)ProcTask500ms,NULL);
2056    HWSqlCreateThread((void*)ProcTask10s,NULL);
2057    HWSqlCreateThread((void*)ProcTask600s, NULL);
2058
2059
2060    while (1)
2061    {
2062        HWSqlDelay(1000);
2063    }
```

```
while (1) {
    //clockBegin=clock();
    gettimeofday(&clockBegin, NULL);
    for (ulConvIndex = 0; ulConvIndex <= 1; ulConvIndex++)
    {
        lRet = HWSqlGetGridLvrtWavePackage(&ulConvIndex, &ulTimeTri); //如果数据库存在变流器的低穿录波数据，返回1;
        //cout << "ConvIndex :"<<ulConvIndex << "    lRet :" << lRet<<endl;
        if (lRet)
        {
            int32 lPointNum = 0;
            double* ptrWaveDataArray = NULL;
            cout << "getting vrt data from database, convIndex= " << ulConvIndex << ", trigger time =" << ulTimeTri
            for (uint32 ulChannelIndex = 1; ulChannelIndex <= 6; ulChannelIndex++)
            {
                cout << "Channel Index " << ulChannelIndex << endl;
                lRet = HWSqlGetGridLvrtWaveSigleChannel(ulConvIndex, ulChannelIndex, &ptrWaveDataArray, &lPointNum)
                if ((lRet > 0) && (lPointNum > 0) && (ptrWaveDataArray != NULL))
                {
                    /*for (i = 0; i < 20; i++)
```

```
//cout <<"10s task is running " << clockBegin<<endl;
fourier_base_initialize();
t = argInit_Unboundedx1_real_T_time(lPointNum, SampleRate);
/*for (i = 0; i < 100; i++)
{
    cout << "time is :" << t->data[i] << endl;
}*/

if (VolType > 0) {                                          //线电压信号
    Ua = argInit_Unboundedx1_real_T(lPointNum, ptrUab);
    Ub = argInit_Unboundedx1_real_T(lPointNum, ptrUbc);
    Uc = argInit_Unboundedx1_real_T(lPointNum, ptrUca);
    Cal_RMSbase(t, Ua, Ub, Uc, ptrUabRMS, ptrUbcRMS, ptrUcaRMS);
    /*for (i = 0; i < 100; i++) {
        cout << "Uab RMS ::" << ptrUabRMS[i] << ",Ubc RMS ::" << ptrUbcRMS[i] << ",Uca RMS ::" << pt

    }*/
    Line2Phase(ptrUab, ptrUbc, ptrUca, lPointNum); //lPointNum
    UnbalanceRate(ptrUabRMS, ptrUbcRMS, ptrUcaRMS, lPointNum, ptrUNbal);
    /*for (i = 0; i < 100; i++) {
        cout << "Unbalance rate is ::" << ptrUNbal[i] << endl;
```

基于模型的风电并网性能在线评价系统设计

- 风电并网性能在线评价系统
- 基于模型的在线评价算法设计及C代码自动生成
- Linux环境下的代码集成
- 总结

基于模型的风电并网性能在线评价系统开发体验

- 无需手写算法功能代码，大大简化了代码实现的工作量，缩短系统软件开发周期；
- 算法测试在模型级实现，简化了代码测试的工作，加速了系统开发过程；
- 可实现快速的算法升级迭代；
- 自动生成代码避免了代码编写错误，提高了在线评价算法服务程序的运行稳定性。

MATLAB EXPO 2021

Thank you

MathWorks®