# 《测绘程序设计实践》

# 报告



课设名称： 导线网平差程序系统设计

专业班级： 测绘1902

姓 名： 苑艺博

学 号： 8211190509

指导老师： 潘林

2020年7月12 日

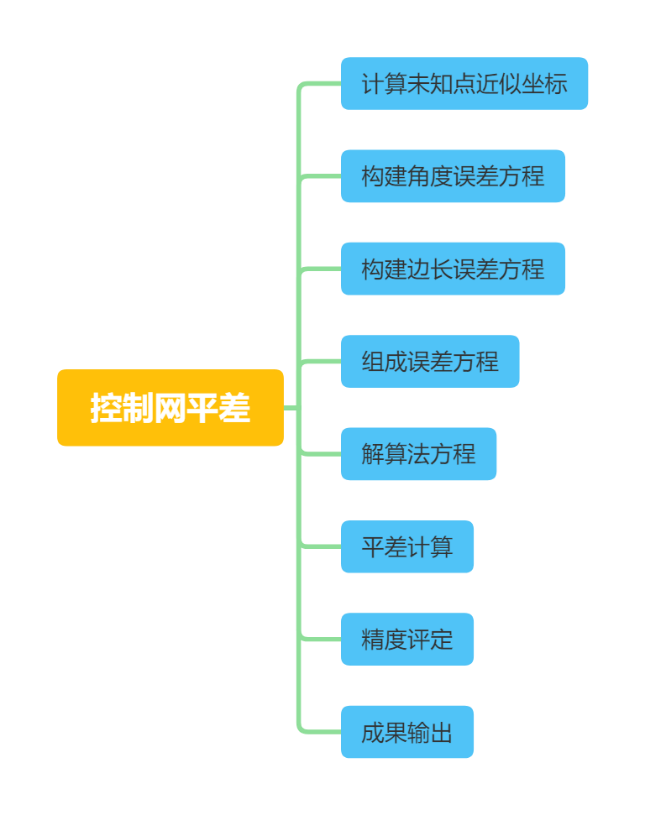
## 需求分析

1. **总体需求**：控制网平差程序对野外控制网观测数据进行平差数据处理，其目的就是根据最小二乘原理，消除网中的各种几何矛盾，求出全网各待定元素（未知点的平面坐标或 三维坐标）。
2. **功能需求**：
3. **优化设计**：根据控制网的观测精度与网形，全面评定网的精度。
4. **数据输入**：文件数据导入。
5. **概算**：未知点近似坐标的推算。
6. **平差计算**：对观测数据进行精密平差计算，得到平差后的点位坐标、方向观测值和边

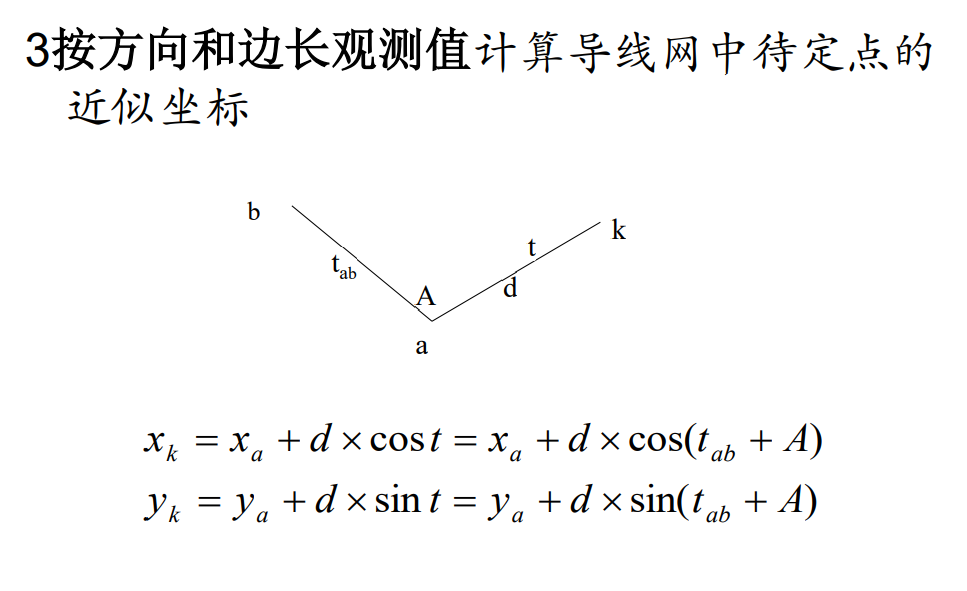
长观测值及相应的改正数，并进行精度评定。

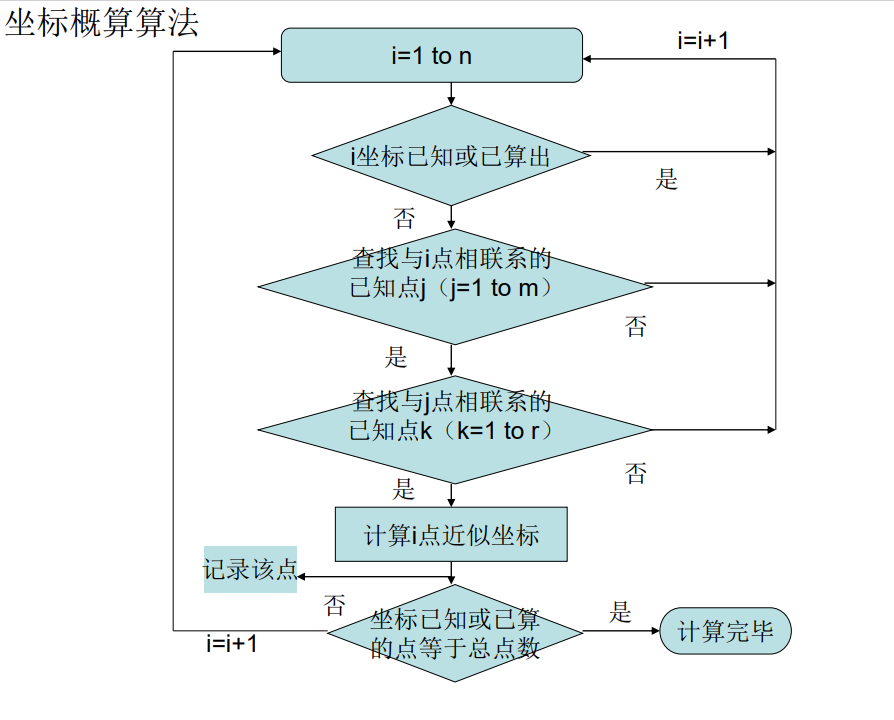
1. **成果输出**：概算结果输出，平差结果报表输出，控制网图形输出， 绘制误差椭圆，绘制坐标轴和比例尺。

## 控制网平差原理



1. **计算未知点近似坐标**
2. 公式：拟采用按方向和边长观测值计算导线网中待定点的近似坐标。





1. 代码实现设想： 这个流程中的关键就是通过哪种联系来求未知点的坐标，主要有两种方法。
2. 第一种办法：通过角度观测值的联系来实现。
3. 遍历角度观测值，由于角度观测值测站点既有已知点，也有未知点，所以角度观测值的测站点为已知点时，可以求出一部分未知点的近似坐标。
4. 接着遍历测站点为未知点的角度观测值。若该测站点的坐标已求出且其零方向的照准点的坐标也已求出，则由该测站点和其零方向照准点的近似坐标可以求出该测站点上其他照准点的坐标。反之，若该测站点的坐标未求出或其零方向的照准点的坐标未求出，则跳过该测站点，遍历下一个测站点。
5. 遍历一遍角度观测值后，判断所有点的近似坐标是否已全部求出，若求出，则停止循环，反之，再次遍历。

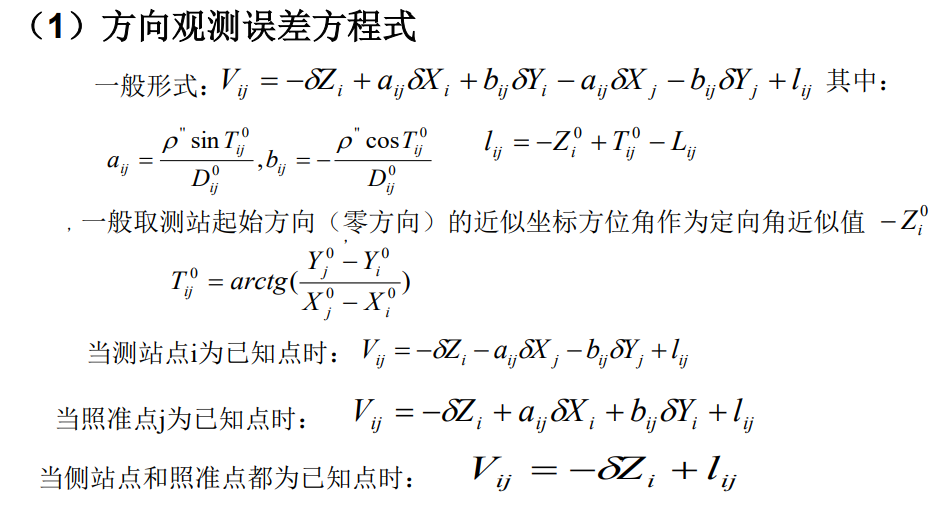
实践证明：该方法虽然简单，但存在一定的缺陷：即测站点和零方向照准点的坐标已知并不能保证该测站上其他照准点的坐标能够求出，因为如果其他照准点和测站点之间不存在边长观测值，这时其他测站点的坐标便求不出来，容错性不佳，虽然可以对此方法继续进行改进，但会增加其复杂度。

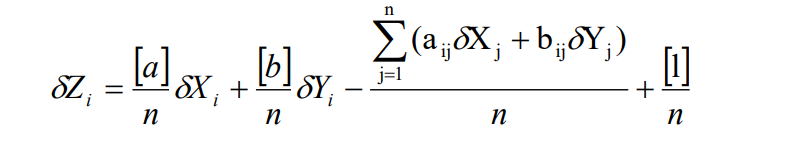
1. 第二种办法：通过边长观测值的联系来实现。

即通过寻找与包含所求未知点的边长，且该边长的另一端点的坐标已求出，然后接着寻找以另一端点为测站点的角度观测值，如果该组角度观测值中存在测站点和照准点坐标均已知的情况，则就可以求出所求未知点的坐标。

实践证明：该方法采用边长的联系来寻找和未知点相联系的已知点，这样在很大程度上可以保证该未知点的坐标可以求出，故容错性更佳。

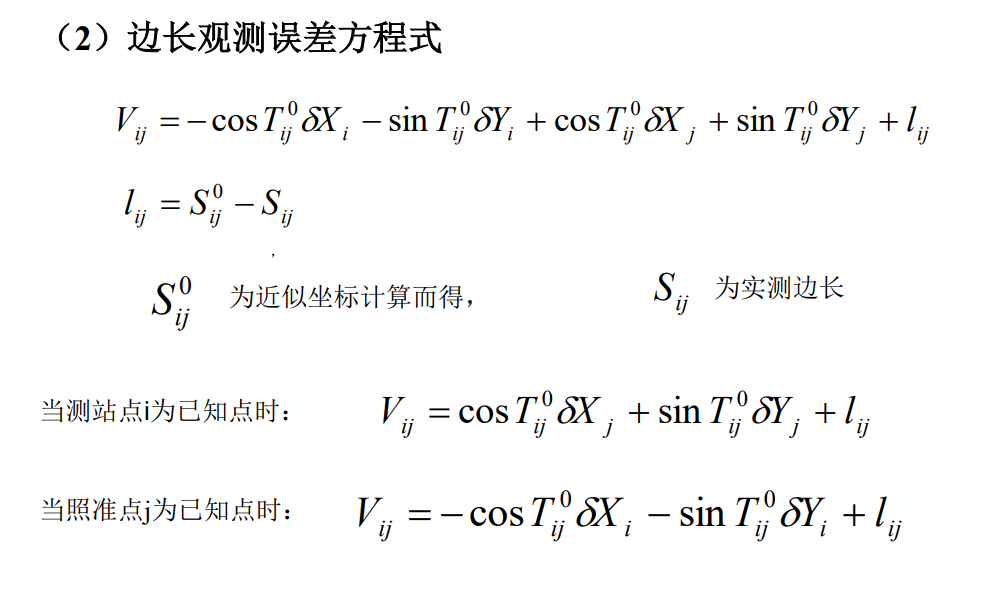
1. **构建角度观测误差方程V=BX+L**
2. 公式：不采用史赖伯一次约化。



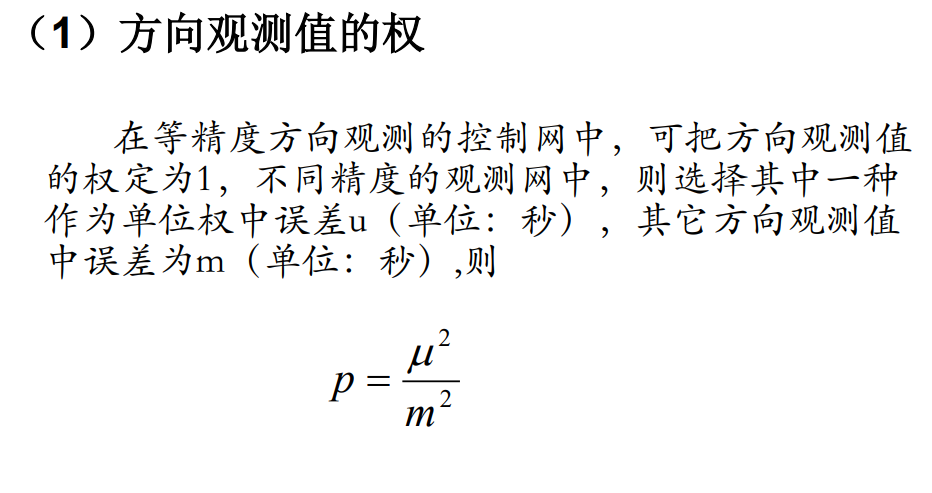


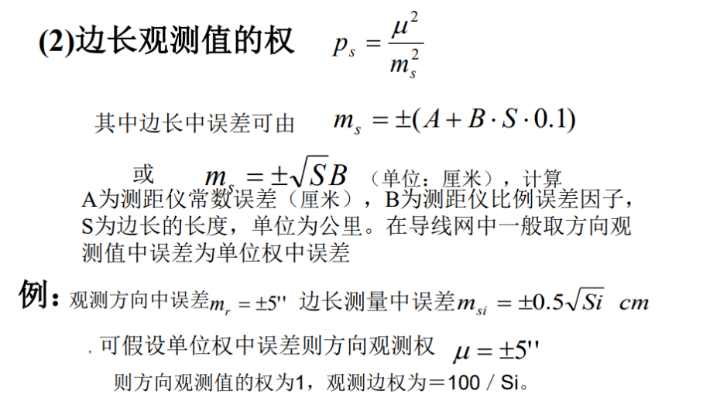
注意：所给aij和bij的公式错误，需要再除以1000

1. 代码实现设想：通过遍历角度观测值来实现。
2. V即各个观测角度，不包括零方向。
3. X即所有未知点的坐标改正数，预设为0，由相应坐标进行替换。
4. B即X的系数，预设为0，然后计算出相应坐标的相应值进行替换。
5. L即常数项，按公式计算即可。
6. **构建边长观测误差方程V=BX+L**
7. 公式：

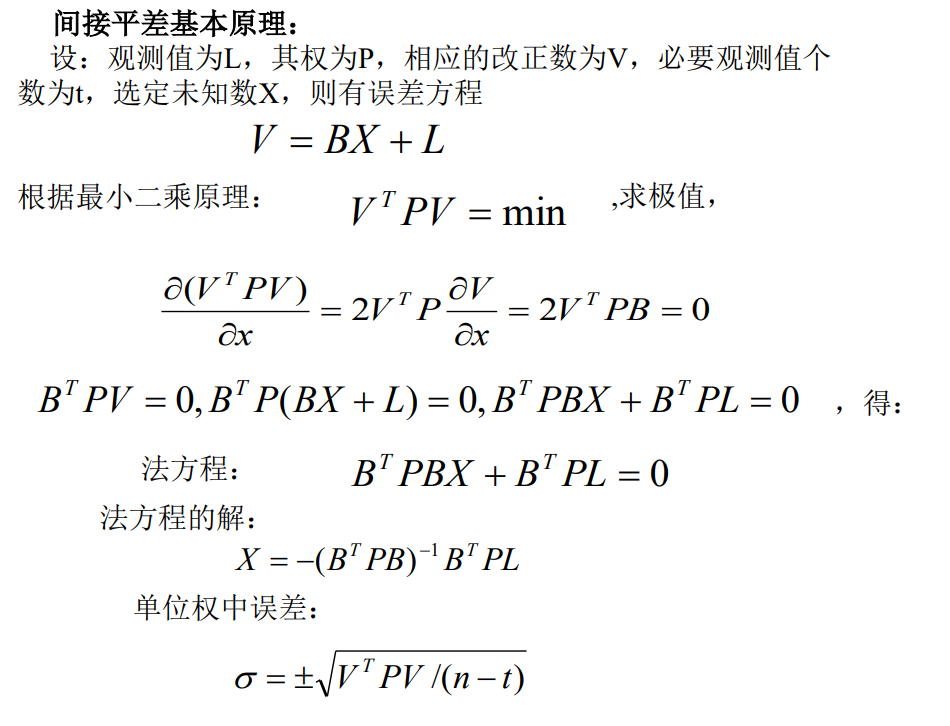


1. 代码实现设想：通过遍历边长观测值来实现。
2. V即各个观测边长。
3. X即所有未知点的坐标改正数，预设为0，由相应坐标进行替换。
4. B即X的系数，预设为0，然后计算出相应坐标的相应值进行替换。
5. L即常数项，按公式计算即可。
6. **组成误差方程**：由角度观测方程和边长观测方程组成总的误差方程V=BX+L。
7. **定权**：包括方向观测值的权和边长观测值的权

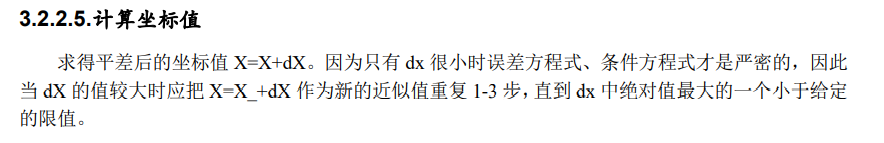




1. **解算法方程**：即求出未知点的改正数X。

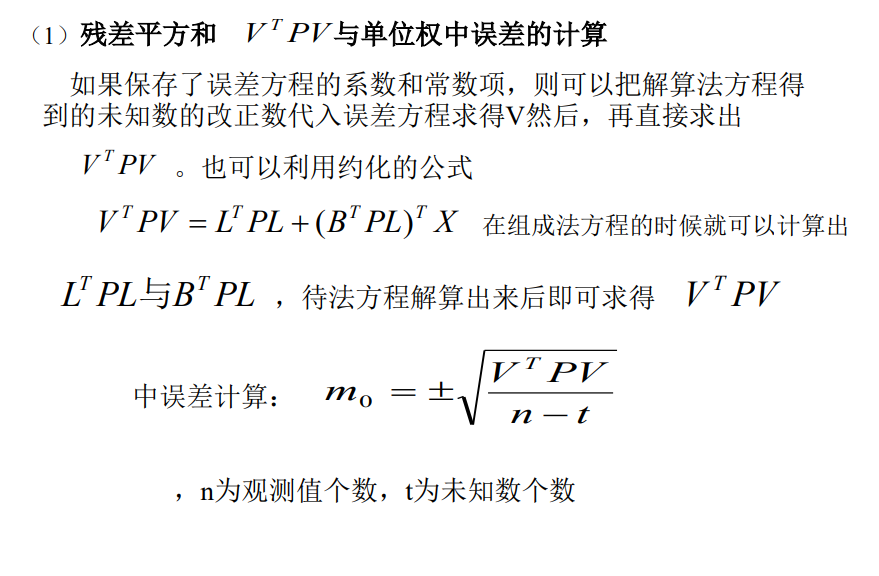


1. **平差计算**：由求出的未知点的改正数X和相应公式计算平差后的点位坐标，方向观测值，边长观测值及相应改正数。

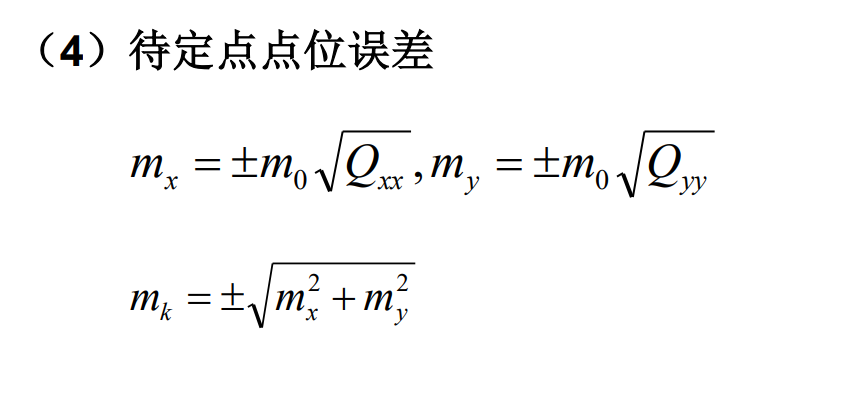


注意：一般dx小于0.1mm即可

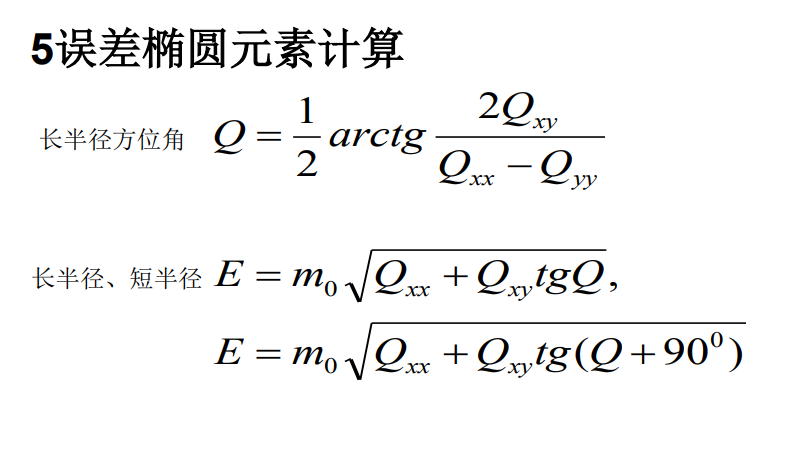
1. **精度评定**：包括单位权中误差σ、点位误差和误差椭圆元素。
2. 单位权中误差σ：



1. 点位误差：



1. 误差椭圆元素：

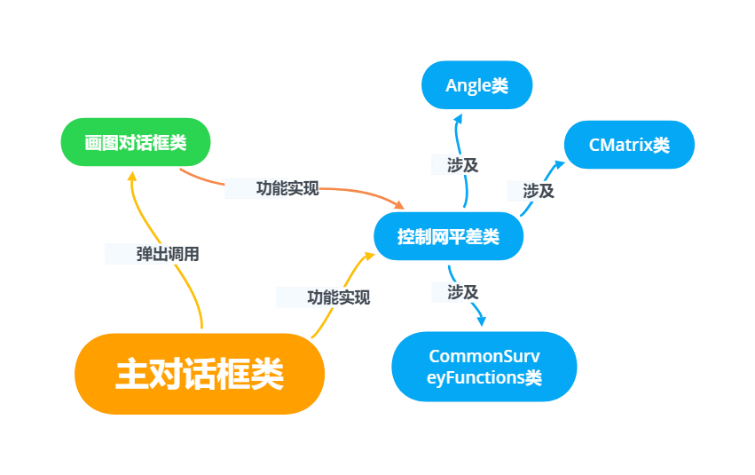
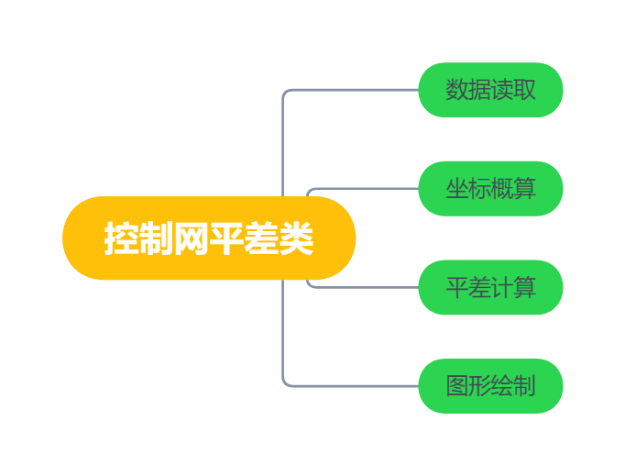


1. **成果输出**：概算结果输出，平差结果报表输出，控制网图形输出， 绘制误差椭圆，绘制坐标轴和比例尺。
2. 概算结果输出：输出未知点的近似坐标。
3. 平差结果报表输出：包括方向观测成果表、距离观测成果表、平面点位误差表和控制点成果表。
4. 绘制控制网网形：由已知点和未知点的平差坐标画出控制网。
5. 绘制误差椭圆：在控制网的基础上绘制误差椭圆。
6. 绘制坐标轴和比例尺：在合适位置绘制坐标轴和比例尺。

### 程序设计与进度完成情况统计

|  |  |  |
| --- | --- | --- |
| 日期（7月） | 任务 | 进度完成情况 |
| 7/5 | 系统总设计 | √ |
| 7/6 | 项目总框架的设计 | √ |
| 7/7 | 计算未知点近似坐标 | √ |
| 7/8 | 角度观测误差方程 | √ |
| 7/9 | 边长观测误差方程 | √ |
| 7/12 | 平差计算 | √ |
| 7/13 | 控制网输出 | √ |
| 7/14 | 平差结果报表输出和绘制误差椭圆 | √ |
| 7/15 | 实验报告书编写和答辩准备 | √ |
| 7/16 | 答辩 | √ |

## 系统架构及模块划分

项目总共包括6个类：主对话框类、画图对话框类、控制网平差类以及涉及的角度类、矩阵类和常用功能类。控制网平差类主要包括三个功能模块：数据输入、坐标概算、平差计算和图形绘制。

## 代码

1. **主对话框类头文件(ControlNetworkDlg.h)的代码：**

|  |
| --- |
| // ControlNetworkDlg.h: 头文件  //  #pragma once  // CControlNetworkDlg 对话框  class CControlNetworkDlg : public CDialogEx  {  // 构造  public:  CControlNetworkDlg(CWnd\* pParent = nullptr); // 标准构造函数  // 对话框数据  #ifdef AFX\_DESIGN\_TIME  enum { IDD = IDD\_CONTROLNETWORK\_DIALOG };  #endif  protected:  virtual void DoDataExchange(CDataExchange\* pDX); // DDX/DDV 支持  // 实现  private:  ControlNetworkAdjust CNA\_Cal;//定义控制网平差类对象  ControlNetworkDrawingDlg CND;//定义控制网绘制对话框类  protected:  HICON m\_hIcon;  // 生成的消息映射函数  virtual BOOL OnInitDialog();  afx\_msg void OnSysCommand(UINT nID, LPARAM lParam);  afx\_msg void OnPaint();  afx\_msg HCURSOR OnQueryDragIcon();  DECLARE\_MESSAGE\_MAP()  public:  friend void ReadFileContent(CStdioFile& sf,CString& strFileContent);//从文件中读取数据到CString字符串中  afx\_msg void OnBnClickedReadfiledata();  CString strData;  CString strOutData;  afx\_msg void OnBnClickedCalapprocoord();  afx\_msg void OnBnClickedAdjustcal();  afx\_msg void OnBnClickedNetdrawing();  }; |

1. **主对话框类cpp文件(ControlNetworkDlg.cpp)的主要代码：**

|  |
| --- |
| //从文件中读取数据到字符串中  void ReadFileContent(CStdioFile& sf,CString& strFileContent)  {  CString strLine;  strFileContent.Empty();  bool bEOF = sf.ReadString(strLine);  while (bEOF)  {  strFileContent += strLine;  bEOF = sf.ReadString(strLine);  if (bEOF) strFileContent += \_T("\r\n");  }  }  //读取数据  void CControlNetworkDlg::OnBnClickedReadfiledata()  {  // TODO: 在此添加控件通知处理程序代码  CFileDialog dlgFile(TRUE, \_T("txt"), NULL,  OFN\_ALLOWMULTISELECT | OFN\_EXPLORER,  \_T("(文本文件)|\*.dat"));  if (dlgFile.DoModal() == IDCANCEL) return;  CString strFileName = dlgFile.GetPathName();  setlocale(LC\_ALL, "chs");  CStdioFile sf;  if (!sf.Open(strFileName, CFile::modeRead)) return;  ReadFileContent(sf, strData);//读取并在文本框中显示文件内容  sf.SeekToBegin();//文件指针定位到开头  CNA\_Cal.ReadFileData(sf);//从文件读取数据  UpdateData(false);  }  //坐标概算  void CControlNetworkDlg::OnBnClickedCalapprocoord()  {  // TODO: 在此添加控件通知处理程序代码  //计算未知点的近似坐标  if (CNA\_Cal.CalUnknownPoint())  {  CNA\_Cal.PrintRoughCalResult(strOutData);//输出概算结果到文本框中  UpdateData(false);  }  }  //平差计算  void CControlNetworkDlg::OnBnClickedAdjustcal()  {  // TODO: 在此添加控件通知处理程序代码  if (CNA\_Cal.AdjustCal())  {  CNA\_Cal.PrintAdjustCalResult(strOutData);//输出平差计算结果到文本框中  UpdateData(false);  }  }  //图形绘制  void CControlNetworkDlg::OnBnClickedNetdrawing()  {  // TODO: 在此添加控件通知处理程序代码  if (CNA\_Cal.JudgeDrawConditon())  {  CND.CNA\_Draw = &CNA\_Cal;//向子对话框中传递控制网平差对象  CND.DoModal();//显示子对话框  }  } |

1. **控制网平差类头文件(ControlNetworkAdjust.h)的代码：**

|  |
| --- |
| #pragma once  #include "ValueName.h"  #include "Matrix.h"  const double EPSILON = 1.0E-12;  const double PI = 4.0 \* atan(1.0);  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \* 类名：ControlNetworkAdjust \*  \* \*  \* 描述：控制网平差类 \*  \* \*  \* 使用方法： \*  \* \*  \* 历史：\*\*日期\*\* \*\*理由\*\* \*\*签名\*\* \*  \* 2021/7/6 无 苑艺博 \*  \* \*  \* 外部类：无 \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  class ControlNetworkAdjust  {  private:  int m\_KnownPointCount;//已知点数  CtrlPoint\* m\_KnownPoint;//已知点指针  int m\_UnknownPointCount;//未知点数  CtrlPoint\* m\_UnknownPoint;//未知点指针  int m\_ObsAngleCount;//观测角度数  ObsAngle\* m\_ObsAngle;//观测角度指针  int m\_ObsDistCount;//观测边长数  ObsDist\* m\_ObsDist;//观测边长指针  //以下是绘图用到的变量  double dXmin;//坐标x的最小值  double dYmin;//坐标y的最小值  double dDetX;//坐标x的最大值和最小值的差值  double dDetY;//坐标y的最大值和最小值的差值  double dScale;//绘图比例  public:  ControlNetworkAdjust();//构造函数  ControlNetworkAdjust(ControlNetworkAdjust& A);//拷贝构造函数  ~ControlNetworkAdjust();//析构函数  private:  friend void ReadFileContent(CStdioFile& sf, CString& strFileContent);//从文件中读取数据到CString字符串中  CAngle Azi(const CtrlPoint& P1, const CtrlPoint& P2);//已知两个控制点，求P1->P2的方位角  double DIST(const CtrlPoint& P1, const CtrlPoint& P2);//已知两个控制点，求P1和P2的距离  int SplitStringArray(CString str, char split, CStringArray& aStr);// 分割字符串，得到字符串数组  CtrlPoint\* SearchPoint(CString strName);//根据点名寻找符合的已知点或未知点，返回CtrlPoint指针变量  bool JudgeUnknownPoint();//判断未知点是否全部求出，若全部求出返回真，反之返回假  ObsAngle\* SearchObsAngle\_StationPtIsLookedPt(CString strName);//寻找测站点为所找点且照准点坐标已求出的角度观测值  ObsAngle\* SearchObsAngle(CString strStationName, CString strObjName);//寻找测站点和照准点的点号符合的角度观测值  ObsDist\* SearchObsDist\_EndPtIsLookedPt(CString strName);//寻找终点为所找点且起点坐标已求出的边长观测值  ObsDist\* SearchObsDist\_StartPtIsLookedPt(CString strName);//寻找起点为所找点且终点坐标已求出的边长观测值  ObsAngle\* SearchZeroDirection(int iNum);//寻找任一角度观测值中测站的零方向所对应的角度观测值  void ObsAngleErrorMatrix(CMatrix& B, CMatrix& L, CMatrix& P);//构建观测角度误差方程和权矩阵  void ObsDistErrorMatrix(CMatrix& B, CMatrix& L, CMatrix& P);//构建观测边长误差方程和权矩阵  void CalBLP(CMatrix& B, CMatrix& L, CMatrix& P);//计算B、L、P  void CalAdjustedAngleDist(CMatrix& V);//计算角度和边长改正数以及改正后边长和角度  void PrecisionEvaluate(CMatrix& Qx,double m0);//精度评定，包括点位误差和误差椭圆参数  void XYCal();//计算dXmin,dYmin,dDetX,dDetY  void SetScale(CRect& rect);//设置自适应窗口的绘图比例  POINT LP2CP(const CtrlPoint& Point, CRect& rect);//从逻辑坐标转换到客户区坐标  void TrianglePointDrawing(CDC\* pDC, const POINT& CentralPoint);//以指定点为中心画小三角形来表示已知点  void PrintCMatrix(CMatrix& A, CString strFileName);//输出矩阵  public:  void ReadFileData(CStdioFile& sf);//从文件中读取数据  bool CalUnknownPoint();//计算未知点近似坐标  void PrintRoughCalResult(CString& strOut);//向文本框中输出概算坐标  bool AdjustCal();//平差计算  void PrintAdjustCalResult(CString& strOut);//向文本框中输出平差坐标，并存入文件中  bool JudgeDrawConditon();//判断是否达到绘图条件  void ControlNetworkDrawing(CDC\* pDC, CRect& rect);//绘制三角网  }; |

1. **控制网平差类cpp文件(ControlNetworkAdjust.cpp)的主要代码：**

|  |
| --- |
| #include "pch.h"  #include "ControlNetworkAdjust.h"  #include "CommonSurveyFunctions.h"  #include <locale>  //构造函数  ControlNetworkAdjust::ControlNetworkAdjust()  {  m\_KnownPointCount = 0;  m\_KnownPoint = NULL;  m\_UnknownPointCount = 0;  m\_UnknownPoint = NULL;  m\_ObsAngleCount = 0;  m\_ObsAngle = NULL;  m\_ObsDistCount = 0;  m\_ObsDist = NULL;  }  //拷贝构造函数  ControlNetworkAdjust::ControlNetworkAdjust(ControlNetworkAdjust& A)  {  m\_KnownPointCount = A.m\_KnownPointCount;  m\_KnownPoint = new CtrlPoint[m\_KnownPointCount];  m\_KnownPoint = A.m\_KnownPoint;  m\_UnknownPointCount = A.m\_UnknownPointCount;  m\_UnknownPoint = new CtrlPoint[m\_UnknownPointCount];  m\_UnknownPoint = A.m\_UnknownPoint;  m\_ObsAngleCount = A.m\_ObsAngleCount;  m\_ObsAngle = new ObsAngle[m\_ObsAngleCount];  m\_ObsAngle = A.m\_ObsAngle;  m\_ObsDistCount = A.m\_ObsDistCount;  m\_ObsDist = new ObsDist[m\_ObsDistCount];  m\_ObsDist = A.m\_ObsDist;  }  //析构函数  ControlNetworkAdjust::~ControlNetworkAdjust()  {  if (m\_KnownPoint != NULL)  {  delete[] m\_KnownPoint;  m\_KnownPoint = NULL;  }  if (m\_UnknownPoint != NULL)  {  delete[] m\_UnknownPoint;  m\_UnknownPoint = NULL;  }  if (m\_ObsAngle != NULL)  {  delete[] m\_ObsAngle;  m\_ObsAngle = NULL;  }  if (m\_ObsDist != NULL)  {  delete[] m\_ObsDist;  m\_ObsDist = NULL;  }  }  //已知两个控制点，求P1->P2的方位角  CAngle ControlNetworkAdjust::Azi(const CtrlPoint& P1, const CtrlPoint& P2)  {  CAngle angAzi;  angAzi(RAD) = Azimuth(P1.dx, P1.dy, P2.dx, P2.dy);  return angAzi;  }  //已知两个控制点，求P1和P2的距离  double ControlNetworkAdjust::DIST(const CtrlPoint& P1, const CtrlPoint& P2)  {  return Dist(P1.dx, P1.dy, P2.dx, P2.dy);  }  //分割字符串  int ControlNetworkAdjust::SplitStringArray(CString str, char split, CStringArray& aStr)  {  int startIdx = 0;  int idx = str.Find(split, startIdx);  aStr.RemoveAll();//先清空  while (-1 != idx)  {  CString sTmp = str.Mid(startIdx, idx - startIdx);  aStr.Add(sTmp);  startIdx = idx + 1;  idx = str.Find(split, startIdx);  }  CString sTmp = str.Right(str.GetLength() - startIdx);  if (!sTmp.IsEmpty())  aStr.Add(sTmp);  return aStr.GetSize();  }  //根据点名寻找符合的已知点或未知点，返回CtrlPoint指针变量  CtrlPoint\* ControlNetworkAdjust::SearchPoint(CString strName)  {  strName.Trim();  CString strPointID;//修整后的点名  //遍历已知点  for (int i = 0; i < m\_KnownPointCount; i++)  {  strPointID = m\_KnownPoint[i].strID.Trim();  if (strName == strPointID)  {  return &m\_KnownPoint[i];  }  }  //遍历未知点  for (int i = 0; i < m\_UnknownPointCount; i++)  {  strPointID = m\_UnknownPoint[i].strID.Trim();  if (strPointID == strName)  {  return &m\_UnknownPoint[i];  }  }  }  //判断未知点是否全部求出，若全部求出返回真，反之返回假  bool ControlNetworkAdjust::JudgeUnknownPoint()  {  for (int i = 0; i < m\_UnknownPointCount; i++)  {  if (m\_UnknownPoint[i].bState == FALSE)  {  return FALSE;  }  }  return TRUE;  }  //寻找测站点为所找点且照准点坐标已求出的角度观测值  ObsAngle\* ControlNetworkAdjust::SearchObsAngle\_StationPtIsLookedPt(CString strName)  {  strName.Trim();  CString strStationPtID;//修整后角度观测值测站点点名  //遍历角度观测值  for (int i = 0; i < m\_ObsAngleCount; i++)  {  strStationPtID = m\_ObsAngle[i].StationPoint->strID.Trim();  if (strName == strStationPtID && m\_ObsAngle[i].ObjPoint->bState == TRUE)  {  return &m\_ObsAngle[i];  }  }  return NULL;  }  //寻找测站点和照准点的点号符合的角度观测值  ObsAngle\* ControlNetworkAdjust::SearchObsAngle(CString strStationName, CString strObjName)  {  strStationName.Trim();  strObjName.Trim();  CString strStationID;//修整后的测站点的点号  CString strObjID;//修整后的照准点的点号  //遍历角度观测值  for (int i = 0; i < m\_ObsAngleCount; i++)  {  strStationID = m\_ObsAngle[i].StationPoint->strID.Trim();  strObjID = m\_ObsAngle[i].ObjPoint->strID.Trim();  if (strStationID == strStationName && strObjID == strObjName)  {  return &m\_ObsAngle[i];  }  }  return NULL;  }  //寻找终点为所找点且起点坐标已求出的边长观测值  //输入：strName为所找点点名  //输出：满足条件的边长观测值的指针变量  ObsDist\* ControlNetworkAdjust::SearchObsDist\_EndPtIsLookedPt(CString strName)  {  CString strEndID;//修整后边长终点点名  strName.Trim();  for (int i = 0; i < m\_ObsDistCount; i++)  {  strEndID = m\_ObsDist[i].EndPoint->strID.Trim();  if (strName == strEndID && m\_ObsDist[i].StartPoint->bState == TRUE)  {  return &m\_ObsDist[i];  }  }  return NULL;  }  //寻找起点为所找点且终点坐标已求出的边长观测值  //输入：strName为所找点点名  //输出：满足条件的边长观测值的指针变量  ObsDist\* ControlNetworkAdjust::SearchObsDist\_StartPtIsLookedPt(CString strName)  {  CString strStartID;//修整后边长终点点名  strName.Trim();  for (int i = 0; i < m\_ObsDistCount; i++)  {  strStartID = m\_ObsDist[i].StartPoint->strID.Trim();  if (strName == strStartID && m\_ObsDist[i].EndPoint->bState == TRUE)  {  return &m\_ObsDist[i];  }  }  return NULL;  }  //寻找任一角度观测值中测站的零方向所对应的角度观测值  //输入：任一角度观测值观测值在数组中的位置iNum  //输出：该角度观测值中测站的零方向所对应的角度观测值，类型为ObsAngle指针变量  ObsAngle\* ControlNetworkAdjust::SearchZeroDirection(int iNum)  {  //向后遍历角度观测值  for (int i = iNum; i >= 0; i--)  {  //如果角度观测值为0，说明该角度观测值所对应的方向即为零方向  if (m\_ObsAngle[i].AngleValue(DMS) == 0)  {  return &m\_ObsAngle[i];  }  }  return NULL;  }  //构建观测角度误差方程和权矩阵  //输入：无  //输出：B、L均为误差方程V=BX+L中角度和边长合矩阵，P为角度和边长合权阵,且在B、L、和P中角度在前，边长在后  void ControlNetworkAdjust::ObsAngleErrorMatrix(CMatrix& B, CMatrix& L, CMatrix& P)  {  double p = 180 / PI \* 3600;//使弧度制转为秒需乘以的常数  //遍历角度观测值，构建观测角度误差方程和权矩阵  for (int i = 0; i < m\_ObsAngleCount; i++)  {  //计算照准方向的方位角近似值  CAngle SightAzi = Azi(\*m\_ObsAngle[i].StationPoint, \*m\_ObsAngle[i].ObjPoint);  //计算零方向的方位角近似值  ObsAngle\* ZeroDirection = SearchZeroDirection(i);  CAngle ZeroAzi;  if (ZeroDirection != NULL)  {  ZeroAzi = Azi(\*ZeroDirection->StationPoint, \*ZeroDirection->ObjPoint);  }  //计算照准方向的边长  double dDist = DIST(\*m\_ObsAngle[i].StationPoint, \*m\_ObsAngle[i].ObjPoint);  //计算误差方程B的系数  double dA = sin(SightAzi(RAD)) / dDist \* p / 1000;  double dB = cos(SightAzi(RAD)) / dDist \* (-p) / 1000;  //计算常数项  CAngle Angle = SightAzi - ZeroAzi;//计算夹角  //如果计算结果小于0  if (Angle(DEG) < 0)  {  Angle(DEG) = Angle(DEG) + 360;  }  CAngle AngleError = Angle - m\_ObsAngle[i].AngleValue;  double dAngleErrorSec = AngleError(DEG) \* 3600;//转为秒  //构建误差方程和权阵  {  //构建B  //如果测站点是未知点  if (m\_ObsAngle[i].StationPoint->ePointStyle == UnknownPoint)  {  //计算各参数在误差方程中B中的列数  int iNumx = 2 \* m\_ObsAngle[i].StationPoint->iNum;//坐标x在误差方程中B中的列数  int iNumy = iNumx + 1;//坐标y在误差方程中B中的列数  B(i, iNumx) = dA;  B(i, iNumy) = dB;  }  //如果照准点是未知点  if (m\_ObsAngle[i].ObjPoint->ePointStyle == UnknownPoint)  {  //计算各参数在误差方程中B中的列数  int iNumx = 2 \* m\_ObsAngle[i].ObjPoint->iNum;//坐标x在误差方程中B中的列数  int iNumy = iNumx + 1;//坐标y在误差方程中B中的列数  B(i, iNumx) = -dA;  B(i, iNumy) = -dB;  }  //构建L  L(i, 0) = dAngleErrorSec;  //构建P  P(i, i) = 1;  }  }  }  //构建观测边长误差方程和权矩阵  //输入：无  //输出：B、L均为误差方程V=BX+L中角度和边长合矩阵，P为角度和边长合权阵,且在B、L、和P中角度在前，边长在后  void ControlNetworkAdjust::ObsDistErrorMatrix(CMatrix& B, CMatrix& L, CMatrix& P)  {  //遍历角度观测值，构建观测角度误差方程和权矩阵  for (int i = 0; i < m\_ObsDistCount; i++)  {  //计算边长的方位角近似值  CAngle DistAzi = Azi(\*m\_ObsDist[i].StartPoint, \*m\_ObsDist[i].EndPoint);  //计算边长近似值  double dDist = DIST(\*m\_ObsDist[i].StartPoint, \*m\_ObsDist[i].EndPoint);  //计算常数项  double dDistErrorMm = (dDist - m\_ObsDist[i].DistValue) \* 1000;//转为毫米  //构建观测边长误差方程和权矩阵  {  //构建B  //如果起点为未知点  if (m\_ObsDist[i].StartPoint->ePointStyle == UnknownPoint)  {  //计算各参数在误差方程中B中的列数  int iNumx = 2 \* m\_ObsDist[i].StartPoint->iNum;//坐标x在误差方程中B中的列数  int iNumy = iNumx + 1;//坐标y在误差方程中B中的列数  B(m\_ObsAngleCount + i, iNumx) = -cos(DistAzi(RAD));  B(m\_ObsAngleCount + i, iNumy) = -sin(DistAzi(RAD));  }  //如果终点为未知点  if (m\_ObsDist[i].EndPoint->ePointStyle == UnknownPoint)  {  //计算各参数在误差方程中B中的列数  int iNumx = 2 \* m\_ObsDist[i].EndPoint->iNum;//坐标x在误差方程中B中的列数  int iNumy = iNumx + 1;//坐标y在误差方程中B中的列数  B(m\_ObsAngleCount + i, iNumx) = cos(DistAzi(RAD));  B(m\_ObsAngleCount + i, iNumy) = sin(DistAzi(RAD));  }  //构建L  L(m\_ObsAngleCount + i, 0) = dDistErrorMm;  //构建P  P(m\_ObsAngleCount + i, m\_ObsAngleCount + i) = 1000 / m\_ObsDist[i].DistValue;  }  }  }  //计算B、L、P  void ControlNetworkAdjust::CalBLP(CMatrix& B, CMatrix& L, CMatrix& P)  {  //计算B中角度观测值零方向的系统误差Z的系数  int iNumz = 2 \* m\_UnknownPointCount;//Z在B中的列数  for (int i = 0; i < m\_ObsAngleCount; i++)  {  //如果是零方向  if (m\_ObsAngle[i].AngleValue(DEG) == 0)  {  B(i, iNumz) = -1;  //把该测站上其他观测值的Z的系数赋值  for (int j = i + 1; j < m\_ObsAngleCount; j++)  {  if (m\_ObsAngle[j].AngleValue(DEG) != 0)  {  B(j, iNumz) = -1;  }  else  {  break;  }  }  iNumz++;  }  }  //计算误差方程和权阵  ObsAngleErrorMatrix(B, L, P);  ObsDistErrorMatrix(B, L, P);  }  //计算角度和边长改正数以及改正后的值  void ControlNetworkAdjust::CalAdjustedAngleDist(CMatrix& V)  {  //计算改正后的角度观测值  for (int i = 0; i < m\_ObsAngleCount; i++)  {  //计算角度改正数,单位秒，保留两位小数  double dAngleError = V(i, 0);  CString strAngleError;  strAngleError.Format(\_T("%.2f"), dAngleError);  m\_ObsAngle[i].AngleError = \_tstof(strAngleError);  //计算改正后的角度值  //由于角度改正数一般不大于60秒，如果大于，则说明观测值存在问题  if (m\_ObsAngle[i].AngleError < 60)  {  m\_ObsAngle[i].AngleValue(DMS) = m\_ObsAngle[i].AngleValue(DMS) + m\_ObsAngle[i].AngleError / 10000;//计算改正后角度值  }  else  {  MessageBox(NULL, \_T("角度改正数大于60秒，角度观测值存在问题，请检查！"), \_T("警告"), MB\_OK);  }  }  //计算改正后的边长观测值  for (int i = 0; i < m\_ObsDistCount; i++)  {  //计算边长改正数，单位m，保留四位小数  double dDistError= V(m\_ObsAngleCount + i, 0) / 1000;  CString strDistError;  strDistError.Format(\_T("%.4f"), dDistError);  m\_ObsDist[i].DistError = \_tstof(strDistError);  //计算改正后的边长  m\_ObsDist[i].DistValue = m\_ObsDist[i].DistValue + m\_ObsDist[i].DistError;  }  }  //精度评定，包括点位误差和误差椭圆参数  void ControlNetworkAdjust::PrecisionEvaluate(CMatrix& Qx, double m0)  {  for (int i = 0; i < m\_UnknownPointCount; i++)  {  //计算未知点坐标x,y在矩阵Qx中的行数  int iNumx = 2 \* m\_UnknownPoint[i].iNum;  int iNumy = iNumx + 1;  double Qxx = Qx(iNumx, iNumx);  double Qyy = Qx(iNumy, iNumy);  double Qxy = Qx(iNumx, iNumy);  //计算点位误差  m\_UnknownPoint[i].dMx = m0 \* sqrt(Qxx);  m\_UnknownPoint[i].dMy = m0 \* sqrt(Qyy);  m\_UnknownPoint[i].dMk = sqrt(m\_UnknownPoint[i].dMx \* m\_UnknownPoint[i].dMx + m\_UnknownPoint[i].dMy \* m\_UnknownPoint[i].dMy);  //计算误差椭圆参数  double dAlfa;//长轴方位角  dAlfa = 0.5 \* atan(2 \* Qxy / (Qxx - Qyy));  if (dAlfa < 0)  {  dAlfa += PI;  }  m\_UnknownPoint[i].dAlfa(RAD) = dAlfa;  m\_UnknownPoint[i].dE = m0 \* sqrt(Qxx + Qxy \* tan(m\_UnknownPoint[i].dAlfa(RAD)));  m\_UnknownPoint[i].dF = m0 \* sqrt(Qxx + Qxy \* tan(m\_UnknownPoint[i].dAlfa(RAD) + PI / 2));  }  }  //计算dXmin,dYmin,dDetX,dDetY  void ControlNetworkAdjust::XYCal()  {  double dXmax, dYmax;  dXmin = m\_KnownPoint[0].dx;  dXmax = m\_KnownPoint[0].dx;  dYmin = m\_KnownPoint[0].dy;  dYmax = m\_KnownPoint[0].dy;  //遍历已知点  for (int i = 0; i < m\_KnownPointCount; i++)  {  dXmin = m\_KnownPoint[i].dx < dXmin ? m\_KnownPoint[i].dx : dXmin;  dXmax = m\_KnownPoint[i].dx > dXmax ? m\_KnownPoint[i].dx : dXmax;  dYmin = m\_KnownPoint[i].dy < dYmin ? m\_KnownPoint[i].dy : dYmin;  dYmax = m\_KnownPoint[i].dy > dYmax ? m\_KnownPoint[i].dy : dYmax;  }  //遍历未知点  for (int i = 0; i < m\_UnknownPointCount; i++)  {  dXmin = m\_UnknownPoint[i].dx < dXmin ? m\_UnknownPoint[i].dx : dXmin;  dXmax = m\_UnknownPoint[i].dx > dXmax ? m\_UnknownPoint[i].dx : dXmax;  dYmin = m\_UnknownPoint[i].dy < dYmin ? m\_UnknownPoint[i].dy : dYmin;  dYmax = m\_UnknownPoint[i].dy > dYmax ? m\_UnknownPoint[i].dy : dYmax;  }  dDetX = dXmax - dXmin;  dDetY = dYmax - dYmin;  }  //设置自适应窗口的绘图比例  void ControlNetworkAdjust::SetScale(CRect& rect)  {  double a = rect.Width();  double b = rect.Height();  double ry = double(rect.Width()) \* 3 / 4 / dDetY;//宽度扩大到2/3，方便在右边画坐标轴和比例尺  double rx = double(rect.Height()) \* 3 / 4 / dDetX;  dScale = rx < ry ? rx : ry;  }  //从逻辑坐标转换到客户区坐标  POINT ControlNetworkAdjust::LP2CP(const CtrlPoint& Point, CRect& rect)  {  POINT P;  P.y = rect.Height() - (Point.dx - dXmin) \* dScale;  P.x = (Point.dy - dYmin) \* dScale;  return P;  }  //以指定点为中心画小三角形来表示已知点  void ControlNetworkAdjust::TrianglePointDrawing(CDC\* pDC, const POINT& CentralPoint)  {  POINT Vertice[3];//三角形三个顶点  Vertice[0].x = CentralPoint.x;  Vertice[0].y = CentralPoint.y - 10;  Vertice[1].x = CentralPoint.x + 8;  Vertice[1].y = CentralPoint.y + 6;  Vertice[2].x = CentralPoint.x - 8;  Vertice[2].y = CentralPoint.y + 6;  for (int i = 0; i < 2; i++)  {  pDC->MoveTo(Vertice[i].x, Vertice[i].y);  pDC->LineTo(Vertice[i + 1].x, Vertice[i + 1].y);  }  pDC->MoveTo(Vertice[0].x, Vertice[0].y);  pDC->LineTo(Vertice[2].x, Vertice[2].y);  }  //绘制控制网  void ControlNetworkAdjust::ControlNetworkDrawing(CDC\* pDC, CRect& rect)  {  XYCal();//计算dXmin,dYmin,dDetX,dDetY  SetScale(rect);//设置自适应窗口的绘图比例  //三角网的原点，可以控制三角网的整体平移  double dOrgNetX = rect.Width() / 4;  double dOrgNetY = -rect.Height() / 5;  CPen pen(PS\_SOLID, 1, RGB(0, 0, 0));  CPen\* pOldPen = pDC->SelectObject(&pen);  POINT StartPoint;//起点在图上的坐标  POINT EndPoint;//终点在图上的坐标  //在已知点上画三角形  StartPoint = LP2CP(m\_KnownPoint[0], rect);  EndPoint = LP2CP(m\_KnownPoint[1], rect);  StartPoint.x = StartPoint.x + dOrgNetX;  StartPoint.y = StartPoint.y + dOrgNetY;  EndPoint.x = EndPoint.x + dOrgNetX;  EndPoint.y = EndPoint.y + dOrgNetY;  TrianglePointDrawing(pDC, StartPoint);  TrianglePointDrawing(pDC, EndPoint);  //根据已知点之间的斜率来画另一条平行线  int dx = StartPoint.x - EndPoint.x;  int dy = StartPoint.y - EndPoint.y;  double dAngle;//垂直于已知点之间直线的直线与X轴的夹角  if (dx == 0)  {  dAngle = PI / 2;  StartPoint.x = StartPoint.x + 3;  StartPoint.y = StartPoint.y;  EndPoint.x = EndPoint.x + 3;  EndPoint.y = EndPoint.y;  }  else  {  dAngle = PI / 2 - atan(dy / dx);  StartPoint.x = StartPoint.x + 3 \* cos(dAngle);  StartPoint.y = StartPoint.y - 3 \* sin(dAngle);  EndPoint.x = EndPoint.x + 3 \* cos(dAngle);  EndPoint.y = EndPoint.y - 3 \* sin(dAngle);  }  pDC->MoveTo(StartPoint);  pDC->LineTo(EndPoint);  //遍历角度观测值，画出照准边  for (int i = 0; i < m\_ObsAngleCount; i++)  {  StartPoint = LP2CP(\*m\_ObsAngle[i].StationPoint, rect);  EndPoint = LP2CP(\*m\_ObsAngle[i].ObjPoint, rect);  StartPoint.x = StartPoint.x + dOrgNetX;  StartPoint.y = StartPoint.y + dOrgNetY;  EndPoint.x = EndPoint.x + dOrgNetX;  EndPoint.y = EndPoint.y + dOrgNetY;  pDC->MoveTo(StartPoint);  pDC->LineTo(EndPoint);  }  //画误差椭圆    //误差椭圆的中心  double dOrgX;  double dOrgY;  //遍历未知点，画误差椭圆  for (int i = 0; i < m\_UnknownPointCount; i++)  {  double dStartX, dStartY, dEndX, dEndY;  //误差椭圆的中心坐标  POINT Org;  Org = LP2CP(m\_UnknownPoint[i], rect);//坐标转换  dOrgX = Org.x + dOrgNetX;  dOrgY = Org.y + dOrgNetY;  double dF = m\_UnknownPoint[i].dF \* 5000;  double dE = m\_UnknownPoint[i].dE \* 5000;  double dAlfa = m\_UnknownPoint[i].dAlfa(RAD);  //绘制短半轴  dStartX = (dF \* sin(dAlfa)) \* dScale + dOrgX;  dStartY = (-dF \* cos(dAlfa)) \* dScale + dOrgY;  dEndX = (-dF \* sin(dAlfa)) \* dScale + dOrgX;  dEndY = (dF \* cos(dAlfa)) \* dScale + dOrgY;  pDC->MoveTo(dStartX, dStartY);  pDC->LineTo(dEndX, dEndY);  //绘制长半轴  dStartX = (-dE \* cos(dAlfa)) \* dScale + dOrgX;  dStartY = (-dE \* sin(dAlfa)) \* dScale + dOrgY;  dEndX = (dE \* cos(dAlfa)) \* dScale + dOrgX;  dEndY = (dE \* sin(dAlfa)) \* dScale + dOrgY;  pDC->MoveTo(dStartX, dStartY);  pDC->LineTo(dEndX, dEndY);  double ex, fy;  ex = dE;  fy = 0;  //转换到长半轴方向上  dStartX = (ex \* cos(dAlfa)  - fy \* sin(dAlfa)  ) \* dScale + dOrgX;  dStartY = (fy \* cos(dAlfa)  + ex \* sin(dAlfa)  ) \* dScale + dOrgY;  pDC->MoveTo(dStartX, dStartY);  for (int i = 6; i <= 360; i += 6)  {  //在坐标轴方向的坐标  ex = dE \* cos((i / 180.0) \* PI);  fy = dF \* sin((i / 180.0) \* PI);  //转换到长半轴方向上  dEndX = (ex \* cos(dAlfa)  - fy \* sin(dAlfa)  ) \* dScale + dOrgX;  dEndY = (fy \* cos(dAlfa)  + ex \* sin(dAlfa)  ) \* dScale + dOrgY;  pDC->LineTo(dEndX, dEndY);  }  }    //画坐标轴  double dLenX = double(rect.Height()) / 10;//坐标轴X轴的长度  double dLenY = double(rect.Width()) / 10;//坐标轴Y轴的长度  //坐标轴的原点坐标  double dOrgCoordX = rect.left + double(rect.Width()) / 10;  double dOrgCoordY = rect.bottom - double(rect.Height()) / 10;  //画Y轴  pDC->MoveTo(dOrgCoordX, dOrgCoordY);  pDC->LineTo(dOrgCoordX + dLenY, dOrgCoordY);  pDC->LineTo(dOrgCoordX + dLenY - 5, dOrgCoordY - 5);  pDC->MoveTo(dOrgCoordX + dLenY, dOrgCoordY);  pDC->LineTo(dOrgCoordX + dLenY - 5, dOrgCoordY + 5);  //画X轴  pDC->MoveTo(dOrgCoordX, dOrgCoordY);  pDC->LineTo(dOrgCoordX, dOrgCoordY - dLenX);  pDC->LineTo(dOrgCoordX - 5, dOrgCoordY - dLenX + 5);  pDC->MoveTo(dOrgCoordX, dOrgCoordY - dLenX);  pDC->LineTo(dOrgCoordX + 5, dOrgCoordY - dLenX + 5);  //画比例尺  //比例尺的起点  double dOrgScaleX = dOrgCoordX + dLenX + double(rect.Width()) / 10;  double dOrgScaleY = dOrgCoordY;  pDC->MoveTo(dOrgScaleX, dOrgScaleY);  pDC->LineTo(dOrgScaleX + 200 \* dScale, dOrgScaleY);  pDC->LineTo(dOrgScaleX + 200 \* dScale, dOrgScaleY - 2);  pDC->MoveTo(dOrgScaleX, dOrgScaleY);  pDC->LineTo(dOrgScaleX, dOrgScaleY - 2);  pDC->SelectObject(pOldPen);  pen.DeleteObject();  CFont Font;//创建字体  VERIFY(Font.CreateFont(  10, // nHeight  0, // nWidth  0, // nEscapement  0, // nOrientation\*  FW\_NORMAL, // nWeight  FALSE, // bItalic  FALSE, // bUnderline  0, // cStrikeOut  ANSI\_CHARSET, // nCharSet  OUT\_DEFAULT\_PRECIS, // nOutPrecision  CLIP\_DEFAULT\_PRECIS, // nClipPrecision  DEFAULT\_QUALITY, // nQuality  DEFAULT\_PITCH | FF\_SWISS, // nPitchAndFamily  \_T("宋体"))); // lpszFacename  //pDC->SetTextColor(RGB(0, 0, 0));///改变字体颜色  CFont\* pOldFont = pDC->SelectObject(&Font);  //写点名  POINT Point;  for (int i = 0; i < m\_KnownPointCount; i++)  {  Point = LP2CP(m\_KnownPoint[i], rect);  Point.x = Point.x + dOrgNetX;  Point.y = Point.y + dOrgNetY;  pDC->TextOutW(Point.x, Point.y + 10, m\_KnownPoint[i].strID);  }  for (int i = 0; i < m\_UnknownPointCount; i++)  {  Point = LP2CP(m\_UnknownPoint[i], rect);  Point.x = Point.x + dOrgNetX;  Point.y = Point.y + dOrgNetY;  pDC->TextOutW(Point.x, Point.y + 10, m\_UnknownPoint[i].strID);  }  //写坐标轴名  pDC->TextOutW(dOrgCoordX - 12, dOrgCoordY - dLenX - 2, \_T("X"));  pDC->TextOutW(dOrgCoordX + dLenY - 3, dOrgCoordY + 5, \_T("Y"));  pDC->SelectObject(pOldPen);  Font.DeleteObject();  //写比例尺名  pDC->TextOutW(dOrgScaleX + 70 \* dScale, dOrgScaleY + 6, \_T("200m"));  }  //从文件中读取数据  void ControlNetworkAdjust::ReadFileData(CStdioFile& sf)  {  CString strLine;  CStringArray aStrTmp;  //读取已知点  sf.ReadString(strLine);  m\_KnownPointCount = \_ttoi(strLine);  m\_KnownPoint = new CtrlPoint[m\_KnownPointCount];  for (int i = 0; i < m\_KnownPointCount; i++)  {  sf.ReadString(strLine);  SplitStringArray(strLine, ',', aStrTmp);  m\_KnownPoint[i].strID = aStrTmp[0];  m\_KnownPoint[i].dx = \_tstof(aStrTmp[1]);  m\_KnownPoint[i].dy = \_tstof(aStrTmp[2]);  m\_KnownPoint[i].bState = TRUE;  m\_KnownPoint[i].ePointStyle = KnownPoint;  }  //读取未知点  sf.ReadString(strLine);  m\_UnknownPointCount = \_ttoi(strLine);  m\_UnknownPoint = new CtrlPoint[m\_UnknownPointCount];  sf.ReadString(strLine);  SplitStringArray(strLine, ',', aStrTmp);  for (int i = 0; i < m\_UnknownPointCount; i++)  {  m\_UnknownPoint[i].strID = aStrTmp[i];  m\_UnknownPoint[i].dx = -1;  m\_UnknownPoint[i].dy = -1;//坐标x、y赋值为-1，方便后面判断是否已经进行概算  m\_UnknownPoint[i].bState = FALSE;  m\_UnknownPoint[i].iNum = i;  m\_UnknownPoint[i].ePointStyle = UnknownPoint;  m\_UnknownPoint[i].dE = -1;//长轴赋值为-1，方便后面判断是否已经平差计算  }  //读取边长观测值  sf.ReadString(strLine);  m\_ObsDistCount = \_ttoi(strLine);  m\_ObsDist = new ObsDist[m\_ObsDistCount];  for (int i = 0; i < m\_ObsDistCount; i++)  {  sf.ReadString(strLine);  SplitStringArray(strLine, ',', aStrTmp);  m\_ObsDist[i].StartPoint = SearchPoint(aStrTmp[0]);  m\_ObsDist[i].EndPoint = SearchPoint(aStrTmp[1]);  m\_ObsDist[i].DistValue = \_tstof(aStrTmp[2]);  }  //读取角度观测值  sf.ReadString(strLine);  m\_ObsAngleCount = \_ttoi(strLine);  m\_ObsAngle = new ObsAngle[m\_ObsAngleCount];  for (int i = 0; i < m\_ObsAngleCount; i++)  {  sf.ReadString(strLine);  SplitStringArray(strLine, ',', aStrTmp);  m\_ObsAngle[i].StationPoint = SearchPoint(aStrTmp[0]);  m\_ObsAngle[i].ObjPoint = SearchPoint(aStrTmp[1]);  m\_ObsAngle[i].AngleValue(DMS)= \_tstof(aStrTmp[2]);  }  }  //计算未知点近似坐标  bool ControlNetworkAdjust::CalUnknownPoint()  {  if (m\_UnknownPointCount == 0)  {  MessageBox(NULL, \_T("未读取数据！"), \_T("警告"), MB\_OK);  return FALSE;  }  else if (m\_UnknownPoint[0].dx >= 0)  {  MessageBox(NULL, \_T("已经完成坐标概算！"), \_T("警告"), MB\_OK);  return FALSE;  }  else  {  do  {  //遍历未知点，计算未知点的近似坐标  for (int i = 0; i < m\_UnknownPointCount; i++)  {  //如果该未知点的坐标未求出  if (m\_UnknownPoint[i].bState == FALSE)  {  ObsAngle\* LookedAngle2 = NULL;//测站点和照准点点均已求出的角度观测值  ObsAngle\* LookedAngle1 = NULL;//测站点坐标已求出且照准点为所求未知点的角度观测值  //寻找终点为未知点且起点坐标已求出的边长观测值  ObsDist\* LookedDist = SearchObsDist\_EndPtIsLookedPt(m\_UnknownPoint[i].strID);    //如果边长观测值存在  if (LookedDist != NULL)  {  //寻找测站点为Dist起点且照准点坐标已求出的角度观测值  LookedAngle2 = SearchObsAngle\_StationPtIsLookedPt(LookedDist->StartPoint->strID);  //寻找照准边为Dist所在边的角度观测值  LookedAngle1 = SearchObsAngle(LookedDist->StartPoint->strID, LookedDist->EndPoint->strID);  }  else  {  //寻找起点为未知点且终点坐标已求出的边长观测值  LookedDist = SearchObsDist\_StartPtIsLookedPt(m\_UnknownPoint[i].strID);  if (LookedDist != NULL)  {  //寻找测站点为Dist起点且照准点坐标已求出的角度观测值  LookedAngle2 = SearchObsAngle\_StationPtIsLookedPt(LookedDist->EndPoint->strID);  //寻找照准边为Dist所在边的角度观测值  LookedAngle1 = SearchObsAngle(LookedDist->EndPoint->strID, LookedDist->StartPoint->strID);  }  }    //如果两个角度观测值存在，证明该未知点坐标可求出  if (LookedAngle1 != NULL && LookedAngle2 != NULL)  {  //计算LookedAngle1和LookedAngle2之间的夹角  CAngle Angle = LookedAngle1->AngleValue - LookedAngle2->AngleValue;  //计算LookedAngle2所在照准边的方位角  CAngle LookedAngle2Azi = Azi(\*LookedAngle2->StationPoint, \*LookedAngle2->ObjPoint);  //计算LookedAngle1所在照准边的方位角  CAngle LookedAngle1Azi = LookedAngle2Azi + Angle;  //如果求得LookedAngle1所在照准边的方位角大于等于360,减去360  if (LookedAngle1Azi(DEG) > 360)  {  LookedAngle1Azi(DEG) = LookedAngle1Azi(DEG) - 360;  }  double dx, dy;//坐标增量  //计算坐标增量  dx = LookedDist->DistValue \* cos(LookedAngle1Azi(RAD));  dy = LookedDist->DistValue \* sin(LookedAngle1Azi(RAD));  //计算照准点的近似坐标  LookedAngle1->ObjPoint->dx = LookedAngle1->StationPoint->dx + dx;  LookedAngle1->ObjPoint->dy = LookedAngle1->StationPoint->dy + dy;  LookedAngle1->ObjPoint->bState = TRUE;  }  }  }  } while (!JudgeUnknownPoint());  return TRUE;  }  }  //输出矩阵到文件  void ControlNetworkAdjust::PrintCMatrix(CMatrix& A, CString strFileName)  {  CStdioFile sf;  if (!sf.Open(strFileName, CFile::modeWrite | CFile::modeCreate)) return;  CString strResult;  CString strContent;  for (int i = 0; i < A.Row(); i++)  {  for (int j = 0; j < A.Col(); j++)  {  strResult.Format(\_T("%-15.4f"), A(i, j));  strContent += strResult;  }  strContent += \_T("\n");  }  sf.WriteString(strContent);  sf.Close();  }  //向文本框中输出概算坐标  void ControlNetworkAdjust::PrintRoughCalResult(CString& strOut)  {  strOut.Empty();  CString strResult;  strOut += (\_T("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*坐标概算\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\r\n\r\n"));  strResult.Format(\_T("%-20s%-22s%-22s\r\n\r\n"), \_T("点号"), \_T("X坐标"), \_T("Y坐标"));  strOut += strResult;  //输出已知点坐标  strOut += (\_T("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*已知点\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\r\n"));  for (int i = 0; i < m\_KnownPointCount; i++)  {  strResult.Format(\_T("%-15s%-20.6f%-20.6f\r\n"), m\_KnownPoint[i].strID, m\_KnownPoint[i].dx, m\_KnownPoint[i].dy);  strOut += strResult;  }  //输出未知点坐标  strOut += (\_T("\r\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*未知点\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\r\n"));  for (int i = 0; i < m\_UnknownPointCount; i++)  {  strResult.Format(\_T("%-15s%-20.6f%-20.6f\r\n"), m\_UnknownPoint[i].strID, m\_UnknownPoint[i].dx, m\_UnknownPoint[i].dy);  strOut += strResult;  }  return;  }  //向文本框中输出平差坐标，并存入文件中  void ControlNetworkAdjust::PrintAdjustCalResult(CString& strOut)  {  CString strFileData;  CString strResult;  strFileData += (\_T("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*平差计算\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"));  //输出方向观测成果  strFileData += (\_T("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*方向观测成果\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"));  strResult.Format(\_T("%-13s%-13s%-13s%-10s%-17s%-10s\n"), \_T("测站"), \_T("照准"), \_T("方向值(dms)"), \_T("改正数(s)"),  \_T("平差后值(dms)"), \_T("备注"));  strFileData += strResult;  for (int i = 0; i < m\_ObsAngleCount; i++)  {  //如果是零方向  if (m\_ObsAngle[i].AngleValue(DEG) == 0)  {  strResult.Format(\_T("%-15s%-15s%-18.6f\n"), m\_ObsAngle[i].StationPoint->strID,  m\_ObsAngle[i].ObjPoint->strID, m\_ObsAngle[i].AngleValue(DMS));  strFileData += strResult;  }  else//如果不是零方向  {  strResult.Format(\_T("%-15s%-15s%-18.6f%-15.2f%-18.6f\n"), m\_ObsAngle[i].StationPoint->strID, m\_ObsAngle[i].ObjPoint->strID,  m\_ObsAngle[i].AngleValue(DMS) - m\_ObsAngle[i].AngleError / 10000, m\_ObsAngle[i].AngleError, m\_ObsAngle[i].AngleValue(DMS));  strFileData += strResult;  }  }  //输出距离观测成果  strFileData += (\_T("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*距离观测成果\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"));  strResult.Format(\_T("%-13s%-15s%-14s%-10s%-13s%-20s\n"), \_T("测站"), \_T("照准"), \_T("距离(m)"), \_T("改正数(m)"), \_T("平差后值(m)"),  \_T("方位角(dms)"));  strFileData += strResult;  for (int i = 0; i < m\_ObsDistCount; i++)  {  CAngle DistAzi = Azi(\*m\_ObsDist[i].StartPoint, \*m\_ObsDist[i].EndPoint);//计算方位角  strResult.Format(\_T("%-15s%-15s%-18.4f%-15.4f%-18.4f%-18.6f\n"), m\_ObsDist[i].StartPoint->strID, m\_ObsDist[i].EndPoint->strID,  m\_ObsDist[i].DistValue - m\_ObsDist[i].DistError, m\_ObsDist[i].DistError, m\_ObsDist[i].DistValue, DistAzi(DMS));  strFileData += strResult;  }  //输出平面点位误差  strFileData += (\_T("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*平面点位误差\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"));  strResult.Format(\_T("%-13s%-12s%-10s%-15s%-16s%-20s\n"), \_T("点名"), \_T("长轴(m)"), \_T("短轴(m)"), \_T("长轴方位角(dms)"),  \_T("点位中误差(m)"), \_T("备注"));  strFileData += strResult;  for (int i = 0; i < m\_UnknownPointCount; i++)  {  strResult.Format(\_T("%-15s%-15.4f%-15.4f%-23.6f%-18.4f\n"), m\_UnknownPoint[i].strID, m\_UnknownPoint[i].dE,  m\_UnknownPoint[i].dF, m\_UnknownPoint[i].dAlfa(DMS), m\_UnknownPoint[i].dMk);  strFileData += strResult;  }  //输出控制点成果  strFileData += (\_T("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*控制点成果\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"));  strResult.Format(\_T("%-19s%-20s%-20s%-13s%-20s\n"), \_T("点名"), \_T("X(m)"), \_T("Y(m)"), \_T("H(m)"), \_T("备注"));  strFileData += strResult;  for (int i = 0; i < m\_KnownPointCount; i++)  {  strResult.Format(\_T("%-15s%-18.4f%-18.4f%-17s%-20s\n"), m\_KnownPoint[i].strID, m\_KnownPoint[i].dx, m\_KnownPoint[i].dy, \_T(" "), \_T("未知点"));  strFileData += strResult;  }  for (int i = 0; i < m\_UnknownPointCount; i++)  {  strResult.Format(\_T("%-15s%-18.4f%-18.4f%-17s%-20s\n"), m\_UnknownPoint[i].strID, m\_UnknownPoint[i].dx, m\_UnknownPoint[i].dy, \_T(" "), \_T(" "));  strFileData += strResult;  }  //输出到文件中  CString strFileName = \_T("AdjustReult.dat");  CStdioFile sf;  if (!sf.Open(strFileName, CFile::modeReadWrite | CFile::modeCreate)) return;  sf.WriteString(strFileData);  MessageBox(NULL, \_T("平差结果已保存到“AdjustReult.dat”文件中！"), \_T("提示"), MB\_OK);  sf.SeekToBegin();//文件指针定位到开头  ReadFileContent(sf,strOut);//从文件读取数据  sf.Close();  }  //判断是否达到绘图条件  bool ControlNetworkAdjust::JudgeDrawConditon()  {  if (m\_UnknownPointCount == 0)  {  MessageBox(NULL, \_T("未读取数据！"), \_T("警告"), MB\_OK);  return FALSE;  }  else if (m\_UnknownPoint[0].dx < 0)  {  MessageBox(NULL, \_T("未坐标概算！"), \_T("警告"), MB\_OK);  return FALSE;  }  else if (m\_UnknownPoint[0].dE < 0)  {  MessageBox(NULL, \_T("未平差计算！"), \_T("警告"), MB\_OK);  return FALSE;  }  else  {  return TRUE;  }  }  //平差计算  bool ControlNetworkAdjust::AdjustCal()  {  if (m\_UnknownPointCount == 0)  {  MessageBox(NULL, \_T("未读取数据！"), \_T("警告"), MB\_OK);  return FALSE;  }  else if (m\_UnknownPoint[0].dx < 0)  {  MessageBox(NULL, \_T("未坐标概算！"), \_T("警告"), MB\_OK);  return FALSE;  }  else if (m\_UnknownPoint[0].dE >= 0)  {  MessageBox(NULL, \_T("已经完成平差计算！"), \_T("警告"), MB\_OK);  return FALSE;  }  else  {  //定义误差方程和权阵  CMatrix X(m\_UnknownPointCount \* 3 + m\_KnownPointCount, 1);  CMatrix B(m\_ObsAngleCount + m\_ObsDistCount, m\_UnknownPointCount \* 3 + m\_KnownPointCount);  CMatrix L(m\_ObsAngleCount + m\_ObsDistCount, 1);  CMatrix P(m\_ObsAngleCount + m\_ObsDistCount, m\_ObsAngleCount + m\_ObsDistCount);  CMatrix V(m\_ObsAngleCount + m\_ObsDistCount, 1);  CMatrix VT;  CMatrix BT;//B的转置矩阵  CMatrix NBB;  CMatrix NBB1;//NBB的逆矩阵  //平差计算直到满足精度要求  double dXmax;//X中坐标x或y增量的最大值  do  {  //计算B、L、P  CalBLP(B, L, P);  BT = ~B;//求B的转置矩阵  NBB = BT \* P \* B;//求NBB  NBB1 = NBB.Inv();//求NBB的逆矩阵  X = -1 \* NBB1 \* BT \* P \* L;//求X  V = B \* X + L;//计算改正数V  //求未知点坐标的近似平差值  for (int i = 0; i < m\_UnknownPointCount; i++)  {  m\_UnknownPoint[i].dx += X(2 \* i, 0) / 1000;  m\_UnknownPoint[i].dy += X(2 \* i + 1, 0) / 1000;  }  //求dXmax  dXmax = fabs(X(0, 0));  for (int i = 0; i < 2 \* m\_UnknownPointCount; i++)  {  dXmax = fabs(X(i, 0)) > dXmax ? fabs(X(i, 0)) : dXmax;  }  } while (dXmax > 0.1);  //将零方向的角度改正归算到该测站其他照准边上  double V0;//记录零方向的角度改正数  for (int i = 0; i < m\_ObsAngleCount; i++)  {  //如果是零方向  if (m\_ObsAngle[i].AngleValue(DEG) == 0)  {  V0 = V(i, 0);  V(i, 0) = 0;//零方向改正数化为0  //如果是该测站上其他照准方向  for (int j = i + 1; j < m\_ObsAngleCount; j++)  {  if (m\_ObsAngle[j].AngleValue(DEG) != 0)  {  V(j, 0) = V(j, 0) - V0;  }  else  {  break;  }  }  }  }  //计算边长和角度观测值的改正数以及改正后的值  CalAdjustedAngleDist(V);  //计算单位权中误差  VT = ~V;  CMatrix VV = VT \* P \* V;  double m0 = sqrt(VV(0, 0) / double(V.Row() - X.Row())) / 1000;  //精度评定，计算点位误差和误差椭圆参数  PrecisionEvaluate(NBB1, m0);  return TRUE;  }  } |

1. **画图对话框类头文件(ControlNetworkDrawingDlg.h)的代码：**

|  |
| --- |
| #pragma once  // ControlNetworkDrawingDlg 对话框  class ControlNetworkDrawingDlg : public CDialogEx  {  DECLARE\_DYNAMIC(ControlNetworkDrawingDlg)  public:  ControlNetworkDrawingDlg(CWnd\* pParent = nullptr); // 标准构造函数  virtual ~ControlNetworkDrawingDlg();  // 对话框数据  #ifdef AFX\_DESIGN\_TIME  enum { IDD = IDD\_CONTROLNETWORKDRAWING\_DIALOG };  #endif  protected:  virtual void DoDataExchange(CDataExchange\* pDX); // DDX/DDV 支持  DECLARE\_MESSAGE\_MAP()  public:  afx\_msg void OnPaint();  //实现  public:  ControlNetworkAdjust\* CNA\_Draw;//控制网平差类对象  }; |

1. **画图对话框类cpp文件(ControlNetworkDrawingDlg.cpp)的主要代码：**

|  |
| --- |
| void ControlNetworkDrawingDlg::OnPaint()  {  CPaintDC dc(this); // device context for painting  // TODO: 在此处添加消息处理程序代码  // 不为绘图消息调用 CDialogEx::OnPaint()  CWnd\* pWin = GetDlgItem(IDC\_CTRLNET);//获取Picture控件的指针  CRect rect;  pWin->GetClientRect(rect);//把控件的长宽、坐标等信息保存在rect里  CDC\* pDC = pWin->GetDC();//获取该控件的画布  CNA\_Draw->ControlNetworkDrawing(pDC, rect);//画图  } |

## 总结

1. **能力考查**：本次课程设计主要考查了我们对控制网平差知识原理的掌握程度以及编程能力，而控制网平差所要求的高精度也需要我们具有缜密的心思和通过调试程序来解决问题的能力，而且，这次课程设计任务量较大、用时较长，更需要我们对软件和系统有整体设计和详细安排。总之，本次课程设计综合提升了我们测绘程序设计的能力。
2. **问题总结**：下面对于程序不同模块的相关问题进行总结。
3. 数据读取：这部分较为简单，但是需要解决一个问题，即对一个文件既要读取数据到事先设置好的数组中，还要输出到对话框中。对此，我先用文件读取函数将文本内容读取到CString变量中，这样便实现了输出到对话框，接着将文本指针重新定位到开头，再逐行读取数据到数组中。这样只需通过文件对话框选择一次文件便可同时满足两种需求。
4. 坐标概算：这部分可采用多种方法，但是每种方法得到的坐标精度可能不同。我尝试了遍历角度、遍历边长和遍历未知点等多种方法，最终得到了一种算法较为简单且精度较高的坐标概算算法。而且，不同的坐标概算算法对数据的规范程度要求不同，如果角度和距离观测值非必要部分缺少，设计的算法也要能够成功进行坐标概算。
5. 平差计算：这部分的难点在于B、L、P矩阵的计算，这需要对相关公式的熟练掌握。同时，这部分也需要解决一个问题，即如何在把平差结果输出到对话框的同时保存到文件中，因为保存到文件时换行符只需\n，而输出到对话框的CString变量时换行符需要\r\n。为了解决这个矛盾，我先将平差结果格式化后保存到文件中，然后将文件指针定位到开头，再通过文件读取函数把存入文件中的平差结果文本输出到对话框中。这样便同时实现了两种功能。
6. 图形绘制：这部分包括三角网形绘制、误差椭圆绘制以及坐标轴和比例尺等元素的绘制，同时还要求各部分的比例大小和相对位置等适中，也要尽可能地满足处理不同数据均能够绘制合适的图形，对画图的程序设计和编程能力要求较高。
7. 用户体验：首先对程序的健壮性和容错性等要求较高，需要应对用户的多种不规范使用行为，比如不按照顺序点击按钮、重复点击按钮以及重复读入数据等等。同时，要求程序界面设计相对简洁美观。而且，要求程序使用方法简单，在不丧失程序必要功能的前提下，尽可能降低程序使用的复杂度。这些要求我自认为处理得不错，基本满足要求。
8. **存在问题**：在定义X矩阵的大小时，行数定义为（3 \* 未知点数 + 已知点数），其实应为（2 \* 未知点数 + 角度测站数），忽略了数据不规范的情况下（未知点数 + 已知点数 != 角度测站数）的情况。
9. **实习收获**：本次课程设计实习，让我对控制网平差相关原理和知识有了更深的掌握，锻炼了我的算法设计、处理bug和代码编写等编程能力，让我认识到测绘程序软件设计不同于其他一般软件设计的特点，即测绘程序设计关键在于对数据的处理，具有较高的精度要求，也让我对测绘程序设计这门学科有了更深的理解和热爱。