University of Alberta

Processing of Daily Agroclimatic Data

by

Darren Paul Griffith

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of
Master of Science

in

Applied Mathematics

Department of Mathematical and Statistical Sciences
Edmonton, Alberta
Spring 2002

Abstract

Soil quality models developed for EcoDistrict polygons (EDP) and the polygons of the Soil Landscapes of Canada (SLC) to monitor the concentration of soil organic matter require daily climate data as input. This thesis (i) provides a method which interpolates the daily station data onto the 149 EDP and 894 SLC polygons, and the 6,900 townships of the province of Alberta, to be used as realistic climate input for soil quality models and drought management, and (ii) describes the implementation of the methods and strategies employed to handle the large amount of data spanning 1 January 1901 to 31 December 2000. The procedure interpolates station data onto a dense network of grid points and then averages the grid point values inside polygons, or assigns them to townships. Two common interpolation methods for spatial data are used, nearest-station assignment and inverse-distance weighting, as well as a hybrid method to handle temporal and spatial variance. The methods are applied to the daily data of maximum temperature, minimum temperature, precipitation, wind speed, wind direction, relative humidity, and total incoming solar radiation, from stations contained within the latitude-longitude box (47°N–64°N, 106°W–124°W). The interpolated data sets and related documentation are available from Alberta Agriculture, Food and Rural Development, Conservation and Development Branch.

Id: thesis.tex,v 6.1 2002/02/09 22:18:06 darren Exp

The wind blows wherever it pleases.
You hear its sound,
But you cannot tell where it comes from
Or where it is going.

John 3:8

Acknowledgements

The results of this thesis were obtained in the pursuit of a Master's degree from the Department of Mathematical and Statistical Sciences at the University of Alberta. I wish to express kind gratitude to my supervisor Professor Samuel Shen, whose leadership and inspiration provided the necessary structure, support, and freedom to complete such a project. I also wish to thank Karen Cannon and Shane Chetner of Alberta Agriculture, Food and Rural Development, Conservation and Development Branch for discussions which led to a better understanding of this work, and for their input as end-users of the data sets.

The Department of Mathematical and Statistical Sciences at the University of Alberta provided a Teaching Assistantship grant to support my study towards this degree.

Further thanks go to Guilong Li for his original work on the project, to Bruce Christensen for ideas on database programming, to Doug Sasaki for preparation of the station data, to Tim Martin and Jilu Feng for discussions on GIS, and to Huamei Yin for her helpful contributions to the station information files and maps.

Contents

1 Introduction **1**

2 Interpolation Methods **8**

2.1 Nearest-station assignment 9

2.2 Inverse-distance weighting 11

2.3 Hybrid method for precipitation frequency 15

3 Background on Data **18**

3.1 Observed data 18

3.2 Processing the station data 34

4 Implementation Details **37**

4.1 Preparation of the station data as input 37

4.2 The single binary data file 39

4.3 Station information files 42

4.4 Interpolation engine outline 45

4.5 Optimization and memory access patterns 50

4.6 Months, leap years, and the use of Julian dates 51

4.7 Output file specifications 53

4.8 Implementation summary 56

5 Results and Accuracy **57**

5.1 Scatter plot method for comparing data sets 57

5.2 Station density 62

5.3 Cross-validation of interpolation methods 68

6 Agricultural Applications and Climate Change 79

A Temporal Interpolations on the Radiation Data 87

B Julian Day Code 89

C Wind Interpolation Code 91

D Temperature Interpolation Code 97

E Shell Script for Manipulating Station Files 103

List of Tables

3.1 Period of record for the seven climate parameters. 20
3.2 Number of stations by parameter grouping and province. . . . 24
3.3 Properties of the processed station data. 34

4.1 A single record in the station data binary file. 40
4.2 Station information file. 43
4.3 Typical run-time parameters for SLC wind interpolation. . . . 46
4.4 Julian day numbers of important dates. 52
4.5 Sizes of interpolated data sets. 55
4.6 Record structure for township output file. 55

5.1 Ratio of observation for each climate parameter. 63
5.2 Errors assessed by cross-validation for Edmonton station. . . . 70
5.3 Sample variances of data from five cross-validation stations. . 72
5.4 Number of days with precipitation per month at Lacombe. . . 73
5.5 Precipitation variances of inverse-distance results and revised results. 74
5.6 Number of precipitation days on two polygons. 75

List of Figures

1.1 The 149 EcoDistrict Polygons of Alberta. 4
1.2 The 894 Soil Landscapes of Canada Polygons of Alberta. . . . 5

2.1 Polygon interpolation diagram. 10

3.1 Location of temperature and precipitation stations. 20
3.2 Location of wind stations. 21
3.3 Location of relative humidity stations. 22
3.4 Location of radiation stations. 23
3.5 Histogram of Canadian daily maximum temperature. 27
3.6 Histogram of U.S. daily maximum temperature. 27
3.7 Histogram of Canadian daily minimum temperature. 28
3.8 Histogram of U.S. daily minimum temperature. 28
3.9 Histogram of Canadian daily precipitation. 29
3.10 Histogram of U.S. daily precipitation. 29
3.11 Histogram of Canadian daily wind speed. 30
3.12 Histogram of U.S. daily wind speed. 30
3.13 Histogram of Canadian daily relative humidity. 31
3.14 Histogram of U.S. daily relative humidity. 31
3.15 Histogram of Canadian total daily incoming solar radiation. . 32
3.16 Histogram of U.S. total daily incoming solar radiation. 32

5.1 Comparison of temperature data for two Canadian stations. . 59

5.2 Comparison of temperature data between a Canadian and a U.S. station. 60
5.3 Ratio of observation for temperature and precipitation, by year. 63
5.4 Histogram of stations at grid points, 1961–1990 temperature. . 65
5.5 Histogram of stations at grid points, 1901–2000 temperature. . 66
5.6 Histogram of stations at grid points, 1901–2000 precipitation. 67
5.7 Precipitation by hybrid interpolation for a major storm. 76
5.8 Precipitation by hybrid interpolation for small, scattered storms. 77

Glossary of abbreviations, agencies, and terms

AAFRD Alberta Agriculture, Food and Rural Development

AES Atmospheric Environment Service, Canada

ASCII American Standard Code for Information Interchange, a standard format for plain text data

CanSIS Canadian Soil Information System

CHU Corn-Heat-Unit, a measurement of growth potential

EDP EcoDistrict Polygons

EPIC Erosion/Productivity Impact Calculator

ET Evapotranspiration

GIS Geographic Information Systems, for displaying spatial data

GRAD Global Radiation, used in U.S. meteorology

GrADS Grid Analysis and Display System, for displaying gridded and station-based meteorological data

ID Identifier, as in "station identifier"

JD Julian day, Julian date, or Julian day number

MAE Mean absolute error

MBE Mean biased error

NOAA National Oceanic and Atmosphere Administration, U.S.A.

NCDC National Climatic Data Center, U.S.A.

NCEP National Centers for Environmental Prediction, U.S.A.

NSDB National Soil DataBase, Agriculture and Agri-Food Canada

NSERL National Soil Erosion Research Laboratory, Purdue University, West Lafayette, Louisiana

period of record The complete record of data for a particular station with respect to time

PET Potential Evapotranspiration

PI Standardized Precipitation Index

RAD Radiation

RF Radiation Field, used in Canadian meteorology

RH Relative humidity

RMSE Root mean square error

SLC Soil Landscapes of Canada

USDA United States Department of Agriculture

WEPP Water Erosion Prediction Project

WMO World Meteorological Organization, Geneva, Switzerland

Chapter 1

Introduction

Agricultural use of climate[1] data has increased considerably during the last two decades due to the rapid development of information technology and the rate of the increase will accelerate in the future (Changnon and Kunkel, 1999). This thesis reports the interpolation methods and implementation used to provide the daily climatic input data required by soil quality models and drought risk management strategies in Alberta, Canada.

Alberta Agriculture, Food and Rural Development (AAFRD), a provincial government department, in partnership with the agricultural industry, has been developing a strategy for sustainable agriculture. AAFRD is committed to environmental sustainability and is working with researchers to develop quantitative measures. Soil quality is one of the initial indicators of environmental sustainability being developed. An aspect of sustainability is to ensure that land management practices maintain or improve soil quality.

Several models are being used by AAFRD to assess soil quality in Alberta. Among them are EPIC (Erosion/Productivity Impact Calculator), and WEPP (Water Erosion Prediction Project). The EPIC model was

[1]The daily atmospheric processes discussed in this thesis are weather and not climate, the latter being long-term averages of weather data. However, in recent usage in agricultural and climatological research they are called *climate data*, which is the convention followed in this thesis.

developed to assess the effect of soil erosion on soil productivity (Sharpley and Williams, 1990). EPIC operates in a daily time step, and requires daily climate data (radiation, maximum and minimum temperature, precipitation, relative humidity, and wind speed) and information on land management practices. The WEPP model, also operating in daily time step, simulates the soil water content in multiple layers of soil relevant to plant growth/decomposition. It also simulates the effects of tillage processes and soil consolidation (Flanagan and Livingston, 1995). These soil quality models apply current knowledge of the crop growth and soil processes which are influenced by climate conditions, to assess the effect of changes in land management practices, such as adoption of reduced tillage or annual cropping or perennial cover, on soil organic matter.

Alberta is developing a method to monitor changes in soil quality by using models on a province-wide scale verified by research plot data. In order to operate, most models require a complete, daily climate data set as input with no missing data. The models quantitatively estimate the effect on soil quality due to changes in land management practices under the climate conditions used in the models. It is very important to have actual, observed daily climate data on which to run the models to compare the model results with carefully measured soil data. Similarly, it is important to have daily climate data for operational use of the models to quantitatively estimate changes in soil organic matter. The daily data must adequately represent the actual weather conditions that occurred. Soil erosion research has shown that severe weather events, especially heavy rainfall or high winds, are primarily responsible for the bulk of soil erosion, which reduces soil quality.

The soil quality models are being developed in Alberta to run on Eco-District polygons (EDP) and Soil Landscapes of Canada (SLC) polygons, whereas the drought monitoring projects require data on Alberta townships. The polygons represent uniform soil and climate conditions, suitable for province-wide land capability assessment and for the soil quality monitoring

intended. The province is divided into 149 EDP polygons (Fig. 1.1) (Agriculture Canada, 1995), 894 SLC polygons (Fig. 1.2) (Shields et al., 1991), and 6,900 townships. There is a mismatch between the climate data available, which have been recorded at points, and the data needed for polygons. It is not clear how a climate parameter for a polygon or a township would be directly measured.

We define the climate value for a polygon or a township as the spatial average of the climate parameter over the area of the polygon or township. Therefore the polygon climate values are calculated quantities. Interpolation and averaging are essential tools used to obtain estimates of these quantities.

Various methods may be used to interpolate scattered data onto polygons, such as the Thiessen polygon method. We chose to interpolate all the available station data onto a regular grid with 10 km spacing and average the values for the grid points inside each polygon. In our current study, the 10 km spacing was chosen as well-suited to the size of the polygons and the station density in Alberta. The grid is not too dense to cause excessive computation. This station-to-grid-to-polygon approach was successfully used earlier by Mackey et al. (1996) to re-characterize climate sub-regions in the province of Ontario, Canada. Since the townships themselves are approximately 10 km by 10 km, the use of a regular grid on this scale is redundant. So the climate parameters were estimated at the centroid of each township, thus serving as the estimate for the whole township itself.

Many methods are available to interpolate data onto grid points, such as nearest-station assignment, inverse-distance weighting, kriging, and thin plate splines. Most of the interpolation methods are best for fitting the mean for a period of a month or longer. However, soil erosion is mostly influenced by extreme weather events, such as heavy precipitation or high winds. Thus, the input data to the soil quality models must adequately represent the real sequence of weather events in the recorded data. According to Karl et al. (1998), the last 90-year's increase of precipitation in the United States

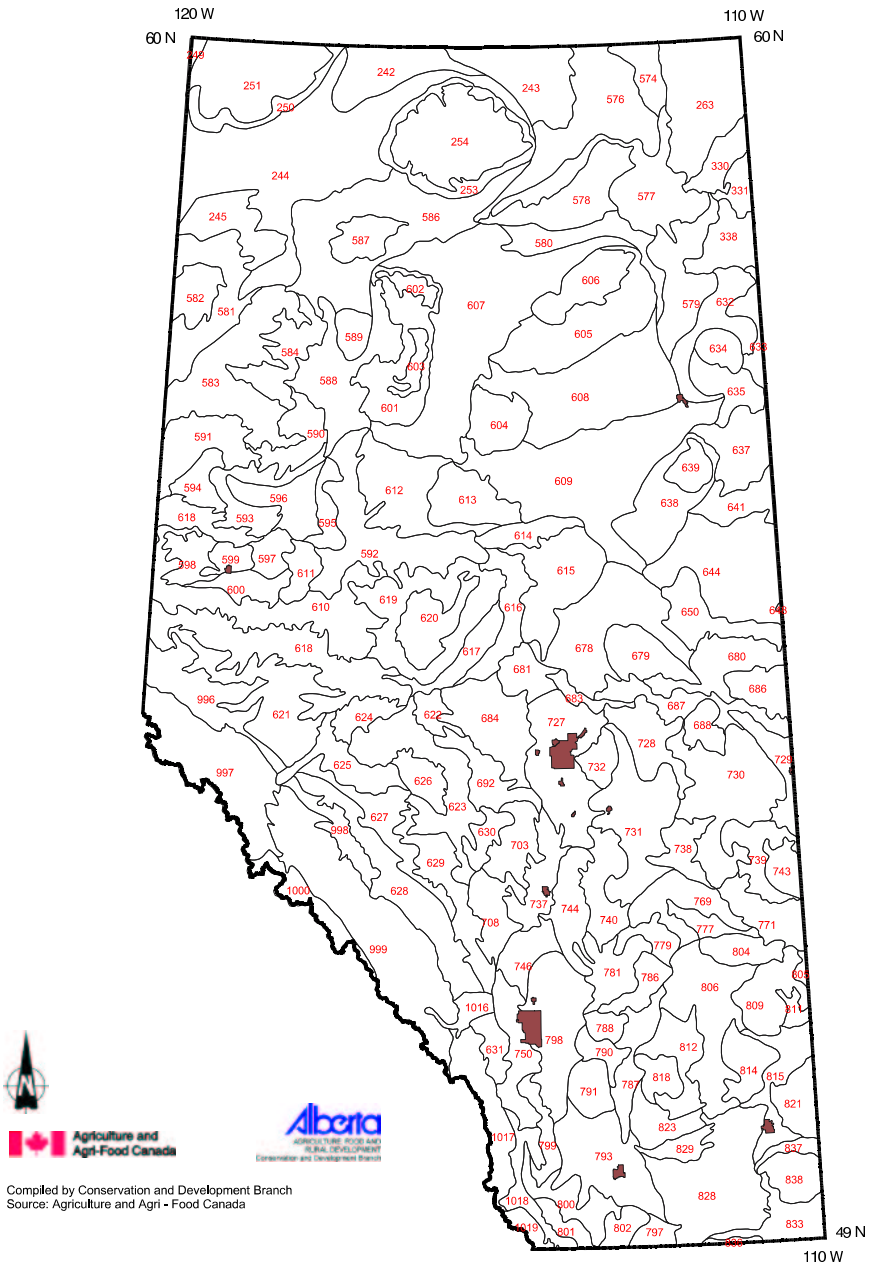

Figure 1.1: The 149 EcoDistrict Polygons of Alberta.

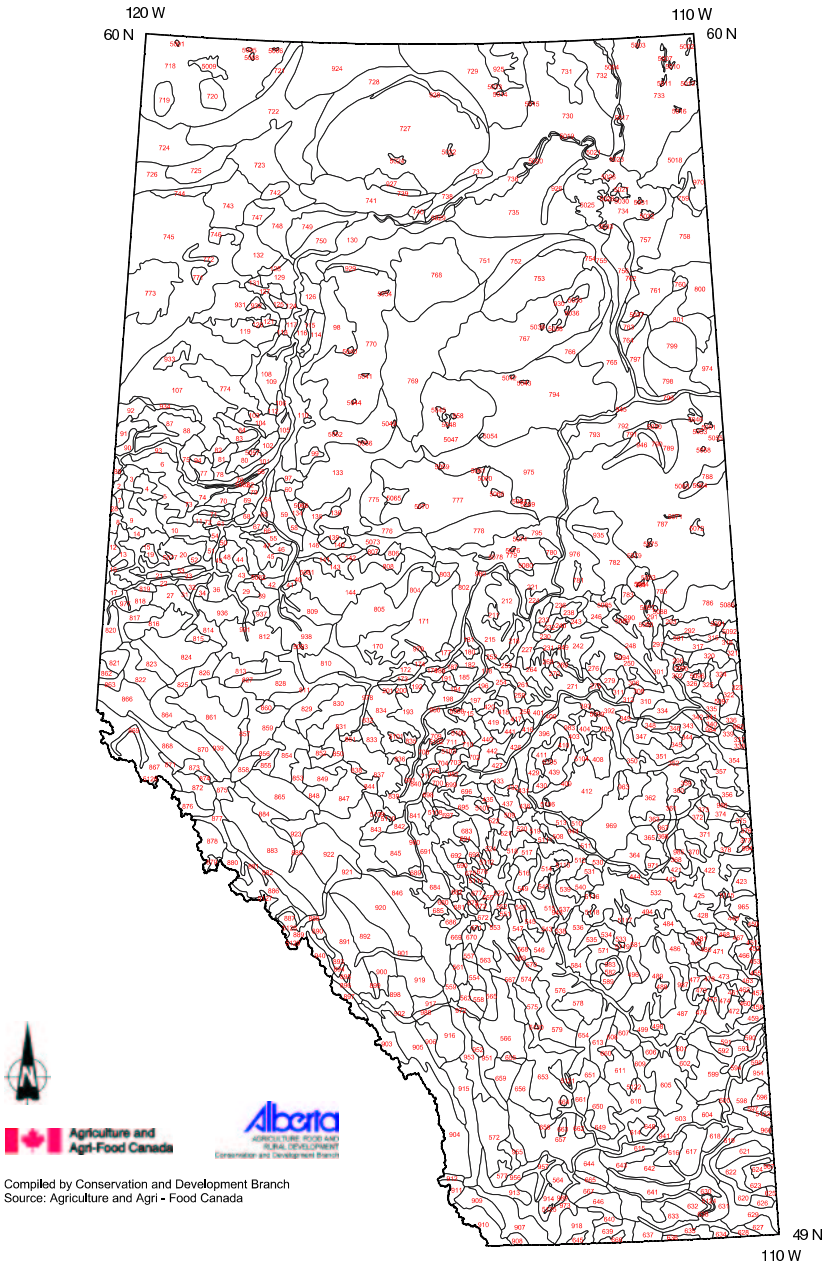

Figure 1.2: The 894 Soil Landscapes of Canada Polygons of Alberta.

and Canada was mainly due to extreme precipitation events. Thus, it is important for the interpolated data to retain the heavy precipitation events.

Preliminary investigation showed that although some interpolation methods provide good estimates of the monthly mean precipitation, they also result in too many days with precipitation and therefore too little precipitation each day. This attribute of interpolation, if left uncorrected, would lead to underestimation of precipitation intensity and hence soil erosion.

Among the commonly used interpolation methods, the nearest-station assignment method yields a good estimate of variance. In order to have a good assessment of climate variability when calculating the EcoDistrict climate normals for 1961–90, Agriculture Canada used the Thiessen polygon approach for interpolation, which is equivalent to the nearest-station assignment method (Bootsma and Ballard, 1999). The details can be found on the website `http://sis.agr.gc.ca/cansis/nsdb/ecostrat/climate_normals_1961-90.html`

This thesis describes the implementation of methods which interpolate the daily station data onto polygons and townships in such a manner that the interpolated data fit not only the monthly means but also retain the appropriate number of days with precipitation. These properties enable the methods to provide realistic and complete (no missing data) daily climate data as input for soil quality models and drought management strategies in Alberta.

Our method for precipitation is a hybrid of inverse-distance weighting and nearest-station assignment. The latter is used to determine whether a polygon has precipitation on a given day; the monthly total precipitation for the polygon, however, is the sum of the daily polygon precipitation determined by the former. The use of data from both methods gives the hybrid method accuracy for monthly totals and the proper variance for daily precipitation amounts. The accuracy of interpolation is assessed by cross-validation for both observed stations and polygons. The cross-validation results show that

our method is reliable and appropriate for preparing realistic daily climate data for use in soil models. To achieve the best fit, we use all of the daily observed climate data available for the period 1 January 1901 to 31 December 2000. Even stations are used that have a very short record, some with as little as two months of data. To overcome the technical difficulty of interpolating data with varying and incomplete data sources, the data are organized so that the interpolation engine works on each day independently from the others and only deals with stations with data on the day being calculated.

Three large data sets, corresponding to EDP polygons, SLC polygons, and townships, are produced using the methods outlined above. Each data set contains complete records of interpolated climate parameters over the time periods for which there is data: 1901–2000 for temperature and precipitation, post-1948 for the others. The EDP data set is 260 MB in size and fits on one CD. Four CDs are required to hold the 1.52 GB of SLC data, which are partitioned according to SLC ID. The township data set is the largest at 5.16 GB, being split into 10 files of 10 years each. These master data sets will be used to provide detailed, continuous, daily climate data of Alberta for soil quality models, ongoing drought monitoring, and a drought indicator study. Use of the data sets will be extended through the calculation of climate normals as well as agricultural derivatives such as degree-days, dew-point-temperature, corn-heat-units, evapotranspiration, frost-free periods, and the like.

This thesis is arranged as follows. Chapter 2 describes the interpolation methods. Chapter 3 describes the data sources used for interpolation. Chapter 4 describes the implementation of the interpolation algorithms in detail. Chapter 5 describes results and the methods used to verify the correctness of the interpolated data, including a discussion of station density. Future investigations and applications of this thesis are discussed in Chapter 6.

Chapter 2

Interpolation Methods

The interpolation methods used in this thesis can be deemed "simple methods" since they require neither the statistical information of the data nor the inversion of a matrix. Thus, "simple" here has three meanings. The method is physically simple because it does not need to consider the statistical or physical properties of the interpolated field. It is mathematically simple because it does not involve any optimization or the complication of linear algebra. It is computationally simple because the computing time required is three orders shorter than that of the sophisticated kriging methods. This simplicity facilitates the understanding of the differences among the results of the various interpolation methods.

Of the many simple methods, two common ones are used here: nearest-station assignment and inverse-distance weighting, described in the following two sections. An evaluation of these methods, and a comparison with other methods used for point estimation in spatial statistics, can be found in Shen et al. (2001). The specific case of polygons will be used here to illustrate the interpolation methods.

The simple methods, although not optimized, can often yield accurate results for daily data because the optimization procedures require an accurate knowledge of a field's covariance structure that is often assumed to be

stationary. However, the daily climate field is usually not stationary. But if monthly or annual data are processed, the complex optimization methods are more accurate.

2.1 Nearest-station assignment

The observed data are from point locations, i.e., climate stations. The desired results are the spatially-averaged values of a climate quantity over each polygon. The ordinary interpolation method yields a value for a point, i.e., the point estimation, while our result ought to be the spatial average over a polygon. To overcome this difficulty, a regular grid of 10 km by 10 km is used to cover the entire area of Alberta. Each grid point is made to correspond with the observational station which is nearest to the grid point, provided that the station has observed data. Furthermore, the grid point is assigned the observed value of the station. Thus, the method is called "nearest-station assignment." Each grid point in a polygon is assigned an observed value. The arithmetic average of the values at all the grid points inside the polygon is the overall value for this polygon, i.e., the climate parameter for the polygon. For example, the polygon P in Fig. 2.1 has six grid points. The value for the polygon P is the arithmetic average of the values assigned to these six points.

In general, because of the chosen grid size, a polygon has at least one grid point. For a small polygon which has no grid points, the centroid of the polygon is selected as the interpolation point. Because the polygon is small compared to the length scales of the seven climatic parameters given in Section 2.2, this centroid represents the climate conditions over the entire small polygon. Three EDP polygons have no regular grid points, (`EDP618`, `648` and `836`), and 116 SLC polygons have no regular grid points. In these cases, their centroids are used as the interpolation points. Thus the regular grid points and the centroids of the small polygons comprise a total of 6,633

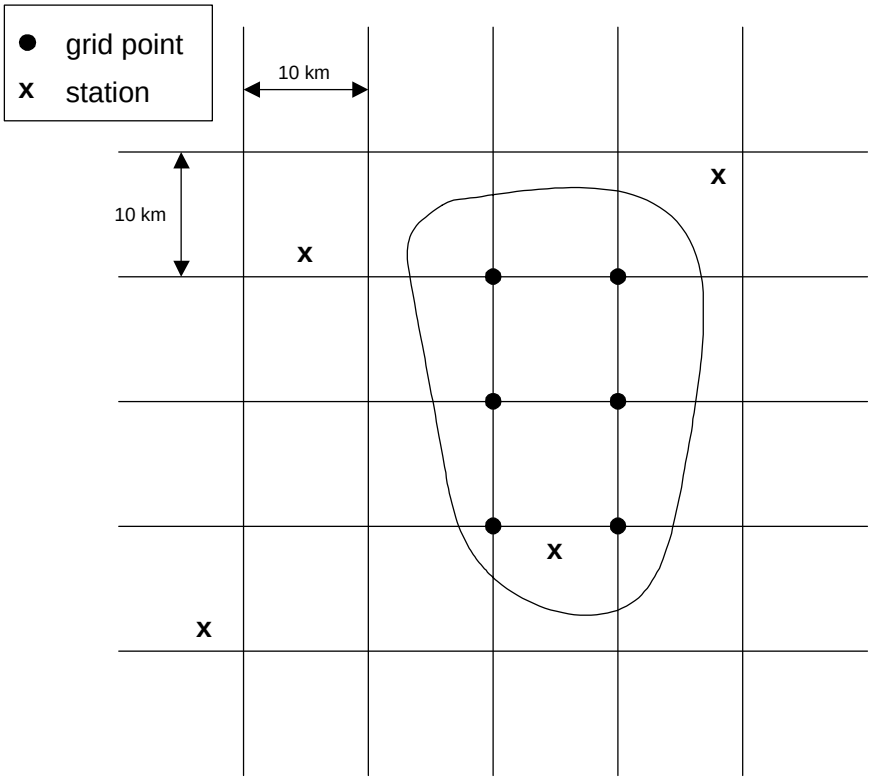

Figure 2.1: Polygon interpolation diagram.

interpolation points for EDP and 6,746 for SLC, generated over the entire area of Alberta. Consequently, this scheme makes it possible for each polygon to have interpolated values of climate parameters assigned to it for any given day.

The nearest-station method uses the climate conditions directly from observed data. It should not yield a large bias when the observational stations are sufficiently dense. However, this method is by no means optimal since no computational optimization is implemented. When the observational stations are very sparse and the climate conditions complex, this method will result in the obvious spatial discontinuity of a climate parameter and perhaps a large interpolation error. The discontinuity occurs across the boundary between two stations, namely the bisector of the line between them.

Since the final result is the spatial average of a parameter on a polygon rather than the grid point values themselves, this interpolation should be related to averaging. An averaging method used in geography is the Thiessen polygon approach, which considers only a station's representative area determined by the bisectors between each pair of stations.[1] As mentioned in the introduction, the Canadian EcoDistrict climate normals for 1961–90, as announced by Agriculture Canada, were obtained by this method. Nearest-station assignment, in fact, is equivalent to the Thiessen polygon method, but the computing for the nearest-station assignment method is much simpler than that of the Thiessen polygon method for daily data (Shen et al., 2001).

2.2 Inverse-distance weighting

This method is based upon the assumption that the influence of nearby observed data on an interpolated point depends solely on the inverse of the

[1]The Thiessen polygon method is also known as Voronoi tessellation and is related geometrically to Delaunay triangulation (Isaaks and Srivastava, 1989).

distance between the interpolated point and the data point. Let $\hat{\mathbf{g}}_j$ be the interpolated point, T_i the observed data at the station $\hat{\mathbf{r}}_i$, and $\hat{T}_j$ the estimated value of the quantity T at the point $\hat{\mathbf{g}}_j$. Then the inverse-distance weighting scheme is

$$\hat{T}_j = \left(\sum_{i=1}^{M_j} \frac{1}{d_{ij}} \right)^{-1} \sum_{i=1}^{M_j} \frac{T_i}{d_{ij}}, \tag{2.1}$$

where $d_{ij} = |\hat{\mathbf{r}}_i - \hat{\mathbf{g}}_j|$ is the distance between $\hat{\mathbf{r}}_i$ and $\hat{\mathbf{g}}_j$, and M_j is the total number of the stations "nearby" $\hat{\mathbf{g}}_j$. If the station $\hat{\mathbf{r}}_i$ is on the grid point $\hat{\mathbf{g}}_j$, then

$$\hat{T}_j = T_i. \tag{2.2}$$

The stations $T_i : (i = 1, 2, \cdots, M_j)$ are chosen according to the distance table for the grid point $\hat{\mathbf{g}}_j$. Station $\hat{\mathbf{r}}_1$ is the station with data that is nearest to the point $\hat{\mathbf{g}}_j$, station $\hat{\mathbf{r}}_2$ is the second nearest, etc. Eight stations with $d_{ij} \leq$ 200 km for temperature data, and with $d_{ij} \leq 60$ km for precipitation data, are chosen for interpolation. Here, the 200 km and 60 km are approximate spatial correlation length scales of temperature and precipitation respectively (Huff and Shipp, 1969; Hansen and Lebedeff, 1987). If fewer than eight stations are within the specified distance, then only the stations present are used for interpolation. For example, if only six stations are within 200 km, then the interpolation for temperature uses only these six stations. In the northern part of Alberta, there may be no stations within the specified distance. In this case, the nearest-station assignment method is used for interpolation since nearest-station assignment does not specify a distance; hence only one station is used. Thus, in data-sparse areas, the inverse-distance method is degenerated into the nearest-station method, but the overall interpolation scheme is so flexible that inside the same polygon, some grid points may use several stations while others may use only a single station.

Our inverse-distance method is somewhat different from the conventional ones described in most geostatistics books or from those commonly used in the literature (Haining, 1990; Isaaks and Srivastava, 1989). The main

differences involve (i) the station searching method and (ii) the use of the nearest-station method if the specified length scales are exceeded in regions with low station density. The spatial length scale is a measure of the coherence of a climate field. If two points are nearby, the correlation coefficient of their climate parameters is close to unity and, in general, the correlation coefficient decreases as the distance between the two points increases. Usually, for a homogeneous field, the length scale of the field is defined as the distance reached when the correlation coefficient is equal to 0.4. Although our highly inhomogeneous daily climate fields do not obey this decaying rule, the length scales are still a good reference for certain coherence. In this thesis, 200 km and 60 km are used as the length scales of temperature and precipitation, respectively. These length scales are approximate values and have large ranges of uncertainties. From the study of Huff and Shipp (1969) on the storm data in Illinois, 60 km is a reasonable value for the daily precipitation length scale. The choice of this value should also consider the station density. If the station density is very large, one may choose 40 km to reduce the noise from distant stations. The length scale for temperature is estimated from the study of Hansen and Lebedeff (1987). They considered annual data. (Monthly data have similar results.) We obtained our length value by dividing theirs, 1,200 km, by $6 \approx \sqrt{30}$, assuming that the daily temperature anomalies are independent from each other. The choice of the length scales has also been validated by examining numerous Alberta daily weather maps.

The use of far away stations most likely introduces more noise when the inhomogeneous teleconnection patterns are not known. Our station searching method inherently adjusts to the station density. The searching method and computational algorithm automatically exclude the more distant stations when there are stations closer to a grid point. Inclusion of a distant station's data, which do not represent the grid point, can only further distort the interpolation result from the true field, which leads to over-smoothing.

The inverse-distance method (2.1) is a point-estimation, but we need the spatial value for each polygon. We used the regular 10 km-by-10 km grid, as in the previous section, to cover Alberta with at least one grid point in each polygon. The inverse-distance method is used for each grid point in a polygon. The climate parameter over the polygon P is the arithmetic average of the $\hat{T}_j$ for all the grid points $\hat{\mathbf{r}}_j$ inside P (see Mackey et al., 1996). Hence,

$$T_P = \frac{1}{N_P} \sum_{j=1}^{N_P} \hat{T}_j. \tag{2.3}$$

If a station is inside the polygon, then some grid points must be near the station, and hence, this station's weight is very large. Thus, if the north-south and east-west dimensions of a polygon are about the same, then the climate values over the polygon are determined mainly by the station(s) inside the polygon. However, if a polygon is long and narrow, a station outside, but near the polygon, may also contribute to the polygon data.

In the data dense region, the field resulting from inverse-distance weighting is smoother than the one from nearest-station assignment, but the inverse-distance weighting might have over-smoothed the field and reduced extremes. The fields of the monthly precipitation and daily mean temperature may be smooth enough, and the inverse-distance weighting may yield reasonable results. For daily precipitation, however, the interpolation method often results in too many days with precipitation in a month, which raises the precipitation frequency of a polygon. For example, if even one of the polygon's nearest stations recorded non-zero precipitation for a given day, inverse-distance weighting will yield a non-zero precipitation for the polygon, even though all other nearest stations for this polygon may actually have recorded zero precipitation for the day. This can significantly increase the number of days with precipitation. For example, the 1961–90 June climatology of `EDP727` has 24.5 days with precipitation using the data from the inverse-distance method, but the days with precipitation for `EDP727` according to our scheme is only 13.7 days. The latter is more reasonable and very similar to the 1961–90 normal

value from actual recorded data at nearby stations. The following section discusses the handling of daily precipitation in detail.

In addition to inverse-distance weighting, one can consider inverse-distance-square weighting, also called the power-2 inverse-distance weighting, which follows the same computational procedures. Due to the higher power of the inverse distance, the field is more localized. Thus, if a polygon has stations inside, the climate parameter over the polygon is subject to little influence from the station data outside of the polygon. Compared to inverse-distance weighting, the inverse-distance-square weighting yields a less smooth climate field due to this localization effect. In fact, as the power-n of the inverse-distance weighting approaches infinity, the power-n inverse-distance weighting becomes the nearest-station assignment.

2.3 Hybrid method for precipitation frequency

A polygon's precipitation frequency involves the polygon's number of precipitation days in a month, the exact days of precipitation, and the amounts precipitated (Osborn and Hulme, 1997). Unfortunately, the inverse-distance method cannot correctly determine the precipitation frequency because the method yields too many wet days and too little precipitation per day for a grid point, and hence for a polygon, compared to the true conditions. The results from the inverse-distance method have smaller variance both in space and time. Since the root mean square error (RMSE), the mean absolute error (MAE), and the mean biased error (MBE), defined by formulas (5.3)–(5.5), are mean properties of differences between the observed data and the interpolated data, it is not surprising that the inverse-distance method yields more accurate results than the nearest-station method when these measures of error are used.

Because the results from the inverse-distance method fit the monthly mean very well, the monthly total computed from the method is accurate.

Our cross-validation for monthly totals for five stations supports this conclusion (see Section 5.3 for the numerical results). Thus, the monthly totals for a grid point—and hence a polygon—are computed from the inverse-distance method.

The only way to achieve a perfect assessment of precipitation frequency is to have stations everywhere on each polygon. Of course, doing so is impossible. In addition, the definition of precipitation over a polygon is not mathematically well-defined. Therefore, as an alternative, we considered that the precipitation of a polygon's centroid indicated whether or not that polygon had precipitation on a given day. Namely, if the centroid observed precipitation, then we concluded that this polygon also had precipitation that day. Usually, the centroid of a polygon was not the location of a station. The centroid's nearest station is the best indication for the centroid's precipitation and hence the polygon's precipitation. Thus, we used the centroid's nearest station as the precipitation indicator of a polygon and also as the variance indicator. The total monthly precipitation of a polygon was computed from the inverse-distance method, and the precipitation frequency was computed from the nearest-station method. For this reason, our method is called a "hybrid method."

For a given day t, the precipitation $Pcpn$ over a polygon was computed from the following empirical and data-driven formula:

$$Pcpn_{edp}(t) = Pcpn_{Iedp}(m)\frac{Pcpn_{centroid}(t)}{Pcpn_{centroid}(m)}, \tag{2.4}$$

where $Pcpn_{Iedp}(m)$ is the monthly total precipitation of the EDP polygon computed from the inverse-distance method, $Pcpn_{centroid}(m)$ is the monthly total precipitation of the station(s) nearest to the centroid, and $Pcpn_{centroid}(t)$ is the precipitation of the station nearest to the centroid for the given day t. In the implementation of this hybrid formula, it is understood that when the denominator $Pcpn_{centroid}(m)$ was zero for a given month m, then $Pcpn_{edp}(t)$ was assigned the value zero.

Please note that in general

$$\sum_{t=1}^{M} Pcpn_{edp}(t) = Pcpn_{Iedp}(m). \tag{2.5}$$

Namely, the monthly total precipitation for a polygon is not changed after using the precipitation frequency formula (2.4)[2]. This hybrid formula combines the inverse-distance and nearest-station methods. It is empirical and data-driven and requires cross-validation, given in Section 5.3.

[2]In the above-mentioned case of $Pcpn_{centroid}(m) = 0$, the monthly total precipitation for a polygon using the hybrid formula will be zero even though the inverse-distance monthly total $Pcpn_{Iedp}(m)$ is not necessarily zero. Hence an exception to the rule (2.5) occurs, however such cases are rare and only a minor concern.

Chapter 3

Background on Data

> Familiarity with the data is an asset in any statistical study. Minor oversights and major conceptual flaws can plague any study. Time spent familiarizing oneself with the data is often rewarded by a quick recognition of when an error has occurred. (Isaaks and Srivastava, 1989).

3.1 Observed data

Selection of stations

The climate stations within Alberta and those 4° of longitude to the east and west, 4° of latitude to the north, and 2° of latitude to the south were used for interpolation. Alberta is contained within the latitude-longitude box (49°N–60°N, 110°W–120°W), with the southwest corner being cut off by the Rocky Mountain Range (Figs. 1.1 and 1.2). Thus our data were from the stations inside the expanded latitude-longitude box (47°N–64°N, 106°W–124°W).

The definition of the boundaries for this box proceeds from a number of factors. First, we wanted a simple definition to distinguish between stations that were appropriate for inclusion in the interpolation process versus those that were not. Because of the inverse-distance length scales applied to tem-

perature and precipitation (200 km and 60 km respectively), stations further than 4° outside of the province would simply not be used. For the U.S. data south of the province, 4° was not tight enough to eliminate the sheer number of stations reporting during the period 1901–2000 (see Fig. 3.1). The reason is that 1° of latitude is approximately 111 km while 1° of longitude is approximately 73 km at 49°N latitude. Hence we reduced the southern boundary of the box to 2°. Furthermore, stations in Washington State were not considered—that is, any U.S. station to the west of 117°W—because of the physical boundary imposed by the Rocky Mountain Range. Station data originating from areas far southwest of Alberta would not properly represent the climate conditions of the southern part of the province due to the influence of the mountains.

Quality control of station data

The raw daily climate data were obtained by AAFRD in various forms and units of measurement from the Atmospheric Environment Service (AES) Canada, and National Oceanic and Atmospheric Administration (NOAA), U.S.A. The observed data were first prepared by Shane Chetner and Douglas Sasaki of AAFRD. Units were standardized, data sets from various agencies were combined, and hourly quantities were totalled and/or averaged into daily quantities. Quality control procedures on the data were performed by both AAFRD and myself before the data were used for interpolation and these will be described in this section. Every piece of raw data, except apparently incorrect data, was used in the interpolation. See Tables 3.1 and 3.2 for a breakdown of stations by climate parameter and location. Note that the large number of stations reported for Yukon, Northwest Territories, and Nunavut is the result of a failure to filter out stations outside the 4° box. But this can be overlooked since the number of stations is small enough not to affect the speed of the interpolation algorithms and furthermore, the data would only be used if no nearer data were available.

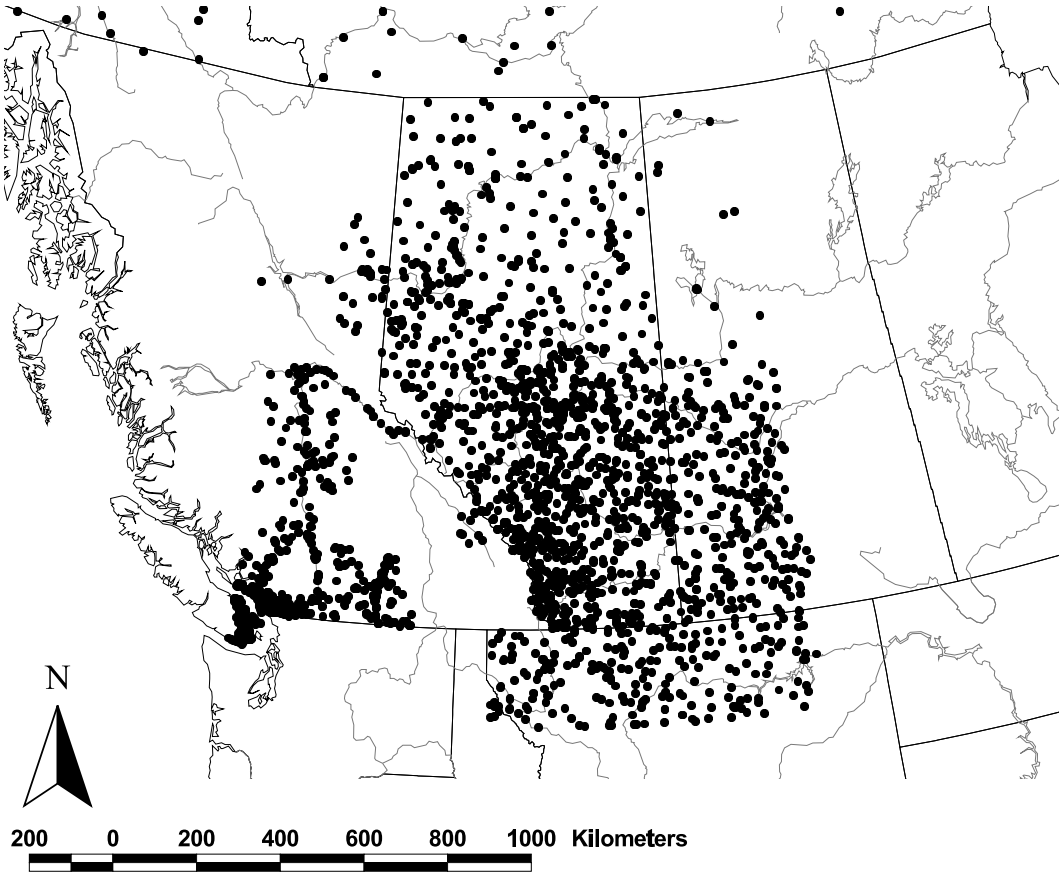

Figure 3.1: Location of temperature and precipitation stations.

Table 3.1: The period of record for the seven daily climate parameters.

Parameter	Start	End	# of Days
Max, Min Temperature			
Precipitation	1 Jan 1901	31 Dec 2000	36,525
RH, Wind	1 Jan 1948	31 Dec 2000	19,359
Radiation	1 Jul 1957	31 Dec 2000	15,890

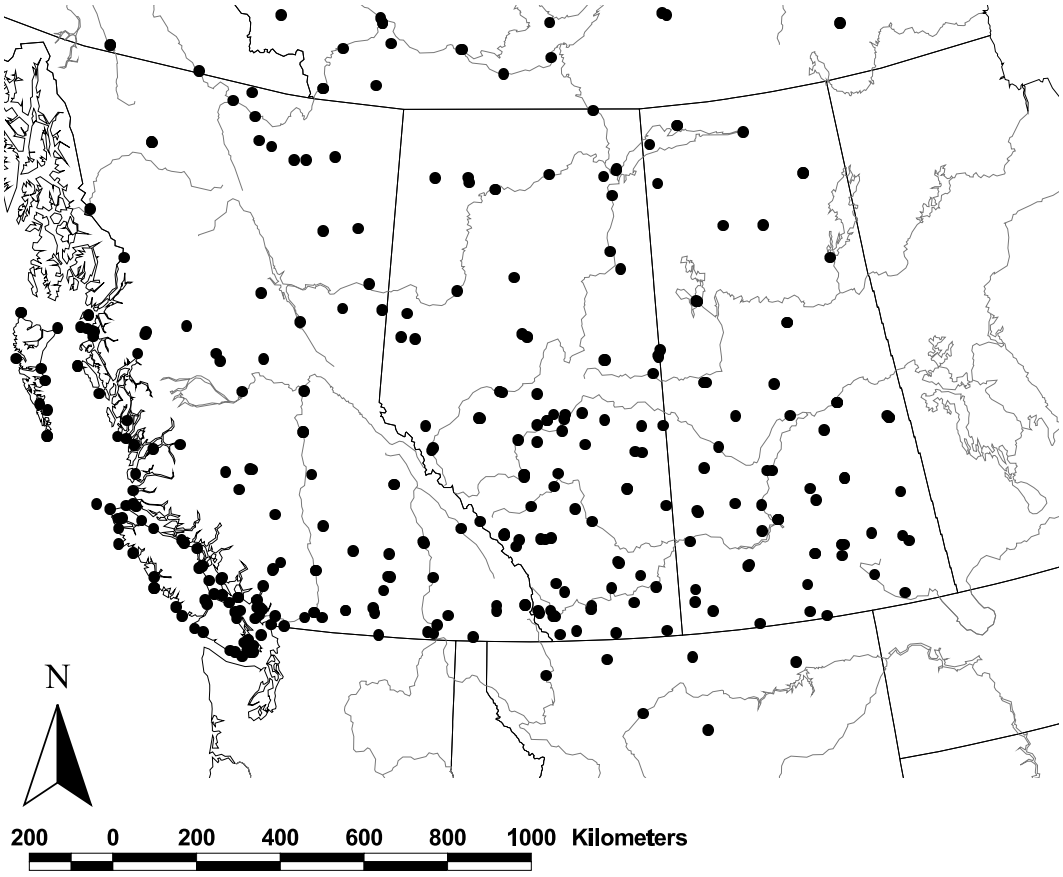

Figure 3.2: Location of wind stations.

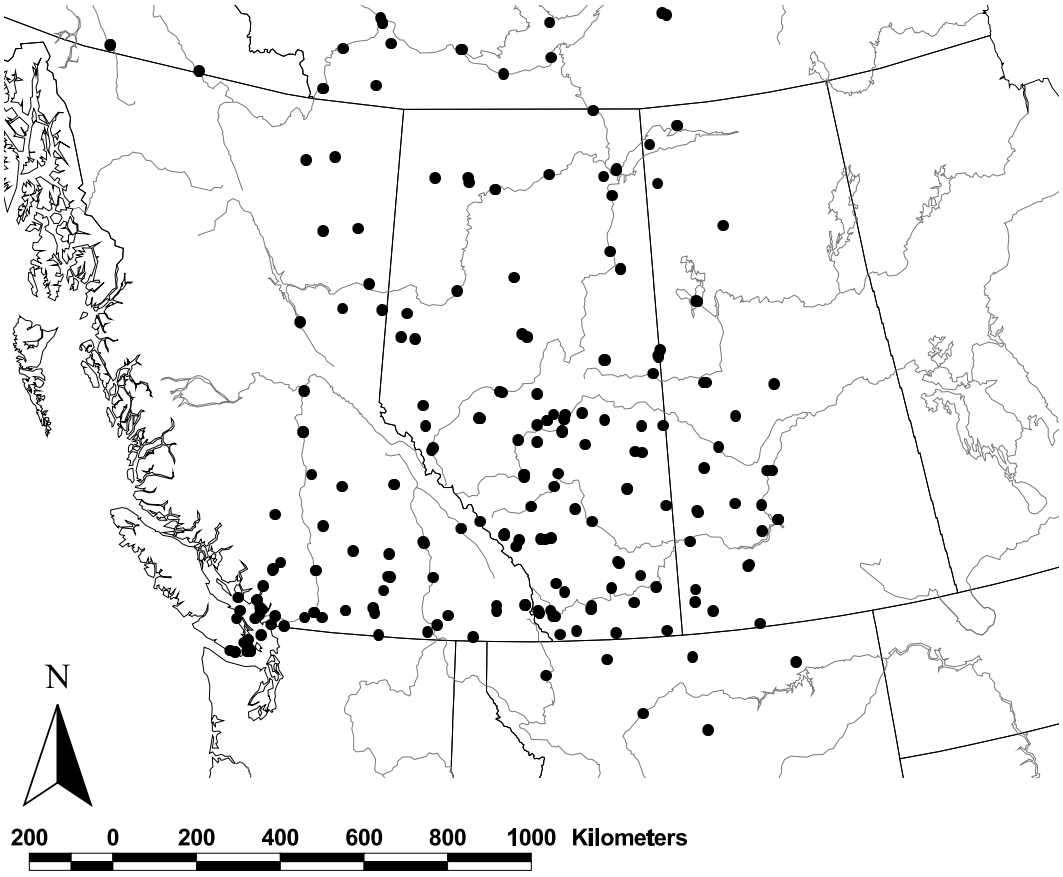

Figure 3.3: Location of relative humidity stations.

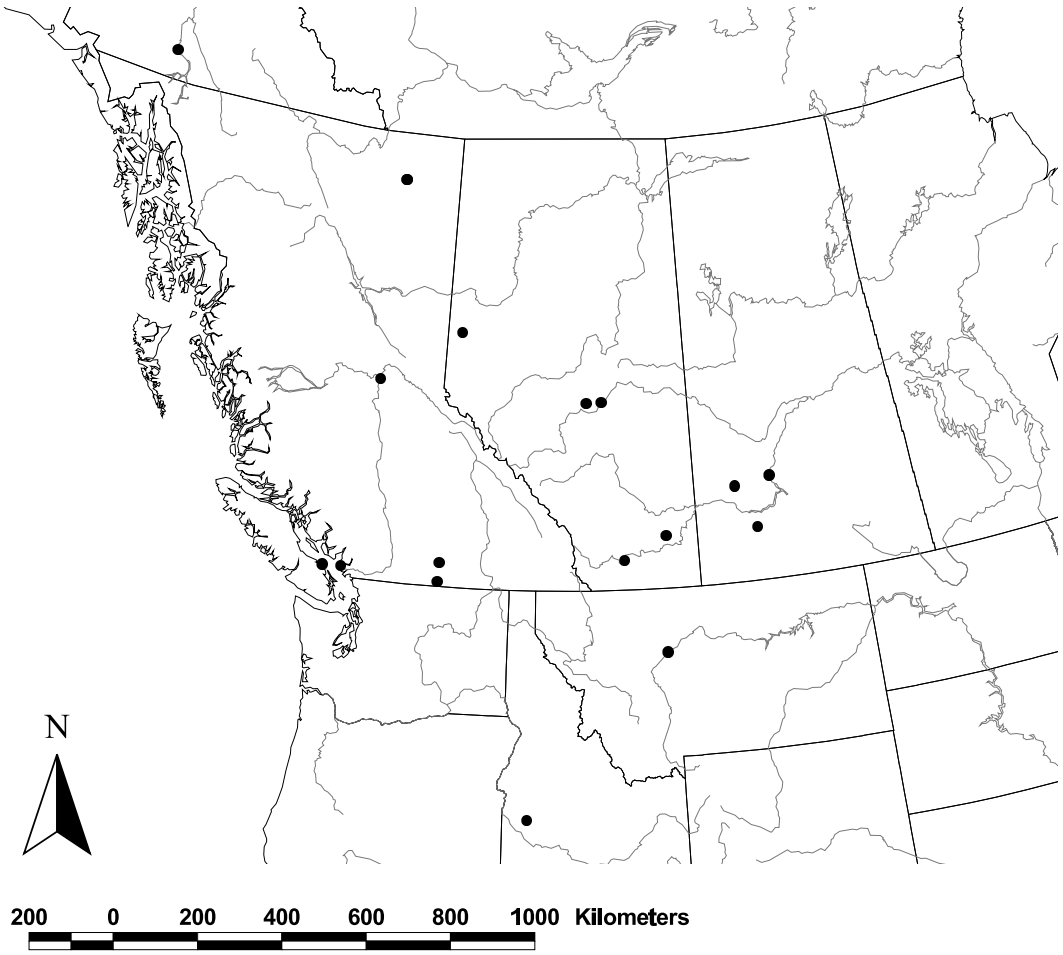

Figure 3.4: Location of radiation stations.

Table 3.2: Number of stations and physical data size for the period 1901–2000, by climate parameter grouping and province. (The abbreviations for the territories stand for Yukon, Northwest Territories, and Nunavut, respectively.)

	Temp Pcpn	Wind	RH	Rad
USA	239	6	6	2
YT, NT, NU	123	188	55	1
SK	283	67	33	3
BC	757	166	74	7
AB	1,209	91	95	5
Canada subtotal	2,372	512	257	16
Total U.S. and Canada	**2,611**	**518**	**263**	**18**
number of lines	14,910,000	2,984,000	1,549,000	125,000
size in kilobytes	626,100	102,000	40,800	3,300

Most of the stations did not have complete data records. Some had missing days and some only a few years of data. For the period of 1901–2000, in the history of all stations, only 12.0% of the days had data for temperature, and 15.2% for precipitation. For any day with precipitation data but not temperature data, or vice-versa, the missing climate parameter was flagged by `-999.9` in the record to indicate this to the user and the interpolation engine. For wind, either a missing direction or speed value would render a record unusable since the interpolation computes wind as a vector quantity. Therefore, both elements would be denoted `-999.9` to signify the removal of the record.

Our quality control strategies include the following. Crude measures were used to eliminate some obviously incorrect data. We searched for the apparently wrong records of maximum temperature less than minimum temperature, temperature greater than 45.0°C and less than -70.0°C, daily precipitation greater than 220.0 mm, relative humidity greater than 100%, daily radiation greater than 36.0 MJ/m^2, wind speed greater than 150.0 km/h, and wind direction greater than 360°. These numbers are based on 1961–90 normals and period-of-record extremes as reported by Environment Canada (1993). An example of a suspect value would be the report of a wind speed of 183.0 km/h on 24 September 1997 by station `2400570`. Since no other stations showed such a high wind speed on the same day, the record is probably not a realistic measurement of the wind conditions on that day. It was set to `-999.9`. Quality control measures such as these prevented some incorrect data from propagating through the interpolation, but the procedures are still somewhat crude. To completely correct or eliminate incorrect station data, better quality control procedures should be adopted in the future.

Computing statistical properties of the station data is useful in ensuring the quality of the data. Histograms provide a tool for visualizing the mean and variance of data and thus facilitate a quick comparison of data sets. Because of the different sources for the station data, it was useful

to construct histograms of the Canadian and the U.S. data separately, to compare their respective characteristics and look for systematic differences between them (cf. Section 5.1). Such a comparison for each climate parameter is shown in the following figures. Figs. 3.5–3.8 display daily maximum and minimum temperature, Figs. 3.9–3.10 precipitation, Figs. 3.11–3.12 wind speed, Figs. 3.13–3.14 relative humidity, and Figs. 3.15–3.16 radiation. From these figures, one can determine the range and frequency of typical values, as indicated by the shape of the histograms.

The temperature histograms, in Figs. 3.5–3.8, show similarities and distinctions between the Canadian and U.S. data. Notice that the Canadian data show more variability (spread). There are two major reasons for this: they cover a larger geographical area than the two-degree latitude band containing the U.S. data, and the higher latitudes of Canada contribute to the larger variance. The means of daily maximum temperature appear to be different, with the Canadian data lower than the U.S. There are more temperatures occurring below 0°C in the Canadian data. For daily minimum temperature, likewise, the Canadian mean is slightly lower, with more values below -20°C and -30°C than for the U.S. data. The larger variance is evident in the Canadian daily minimum temperature histogram also.

Canadian stations record larger amounts of daily precipitation than U.S. stations, as shown in Figs. 3.9–3.10. Events above 10 mm are more common in the Canadian data.

U.S. wind speeds have a larger mean and smaller variance than Canadian wind (Figs. 3.11–3.12). Speeds below 10 km/h are more prevalent in the Canadian data, as are those above 50 km/h, but the higher values have less effect on the mean due to their low population counts.

Relative humidity shows a skewness in the Canadian data towards higher humidity values (Figs. 3.13–3.14). The Canadian average is 80%; the U.S. average is 61%. The U.S. data are more symmetric in shape and have a higher variance.

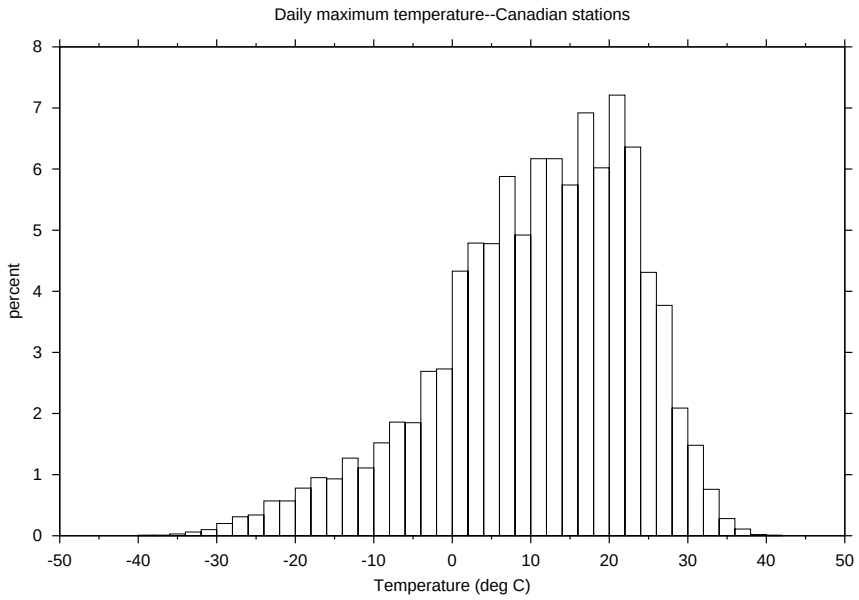

Figure 3.5: Histogram of Canadian subset of daily maximum temperature.

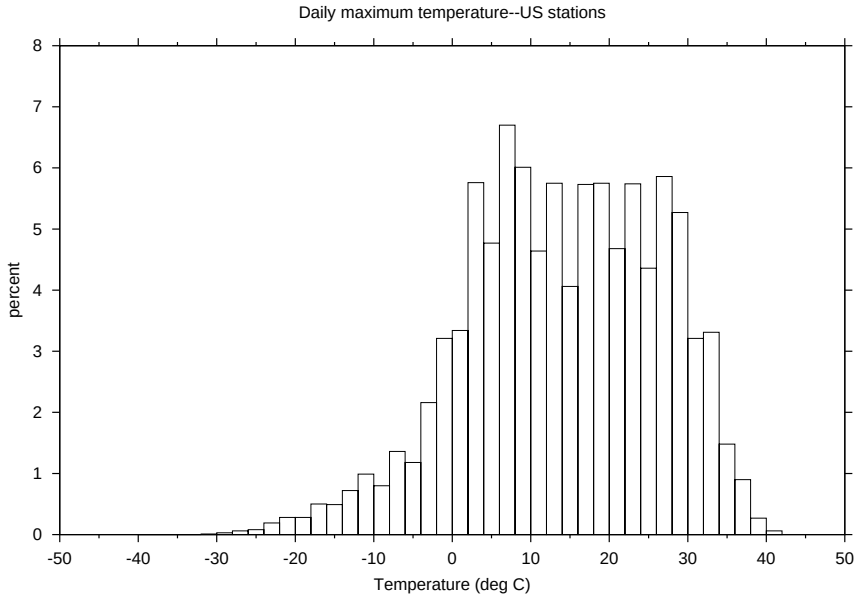

Figure 3.6: Histogram of U.S. subset of daily maximum temperature.

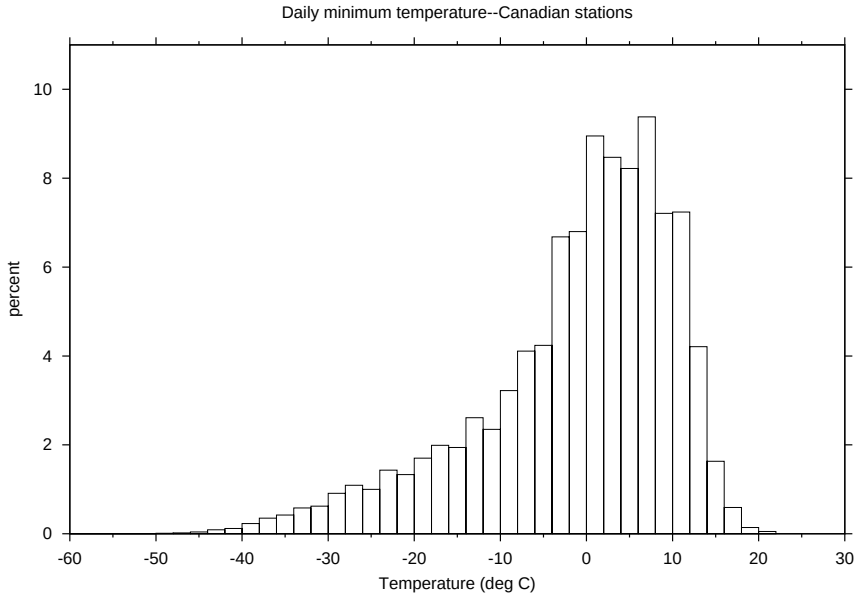

Figure 3.7: Histogram of Canadian subset of daily minimum temperature.

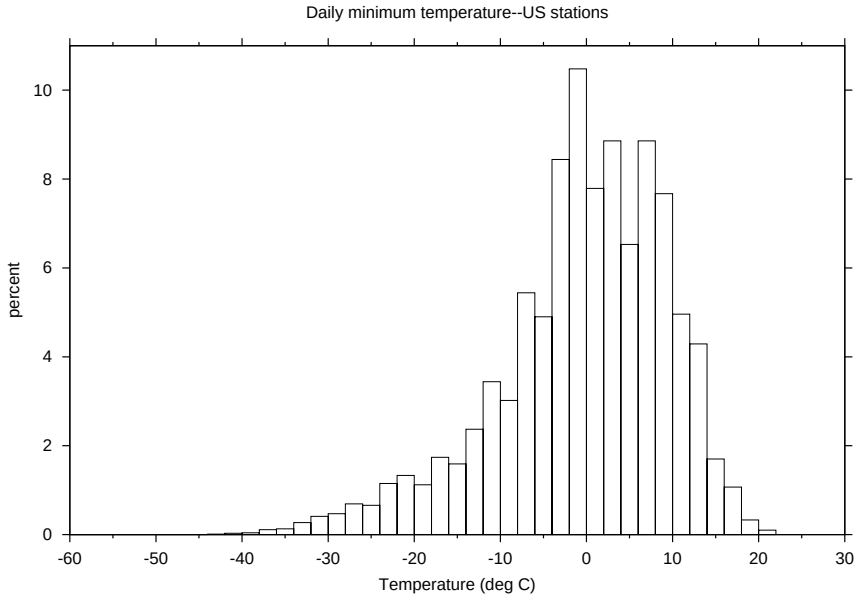

Figure 3.8: Histogram of U.S. subset of daily minimum temperature.

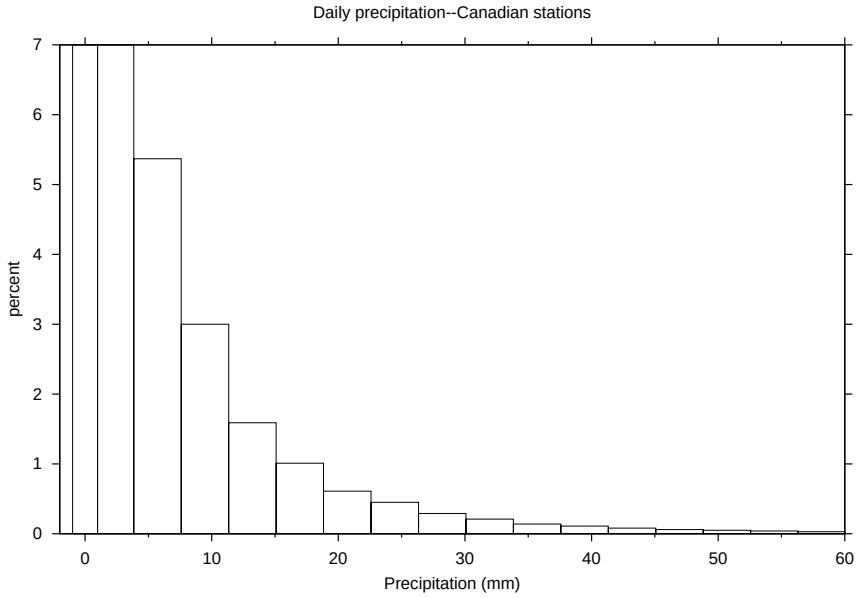

Figure 3.9: Histogram of Canadian subset of daily precipitation.

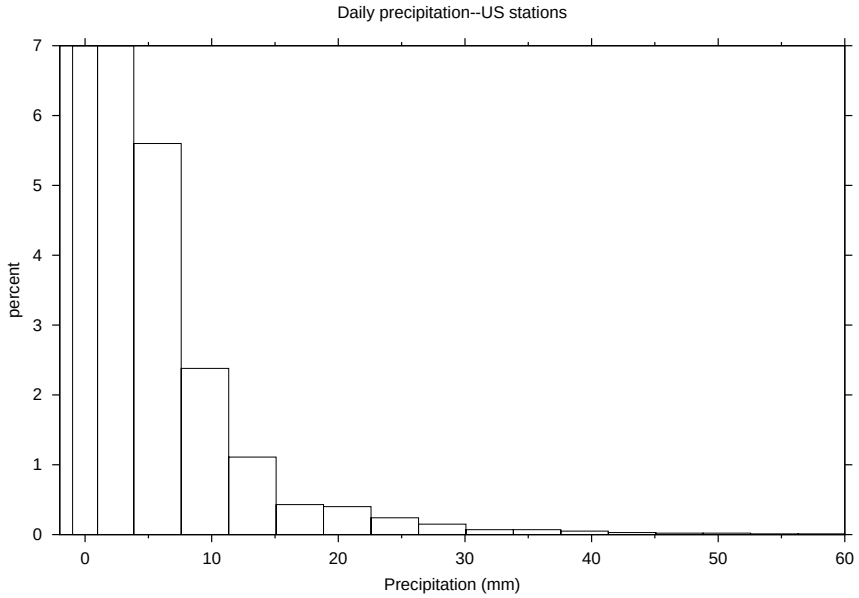

Figure 3.10: Histogram of U.S. subset of daily precipitation.

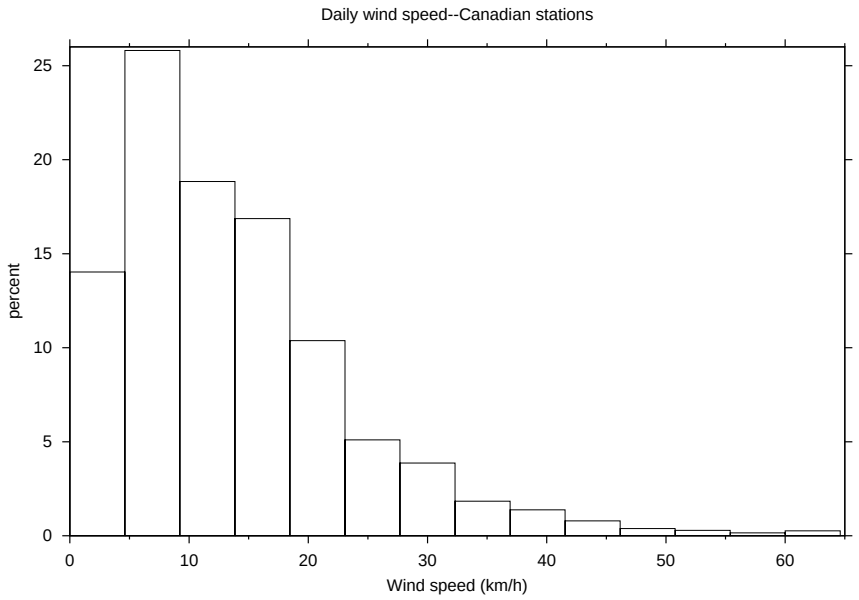

Figure 3.11: Histogram of Canadian subset of daily wind speed.

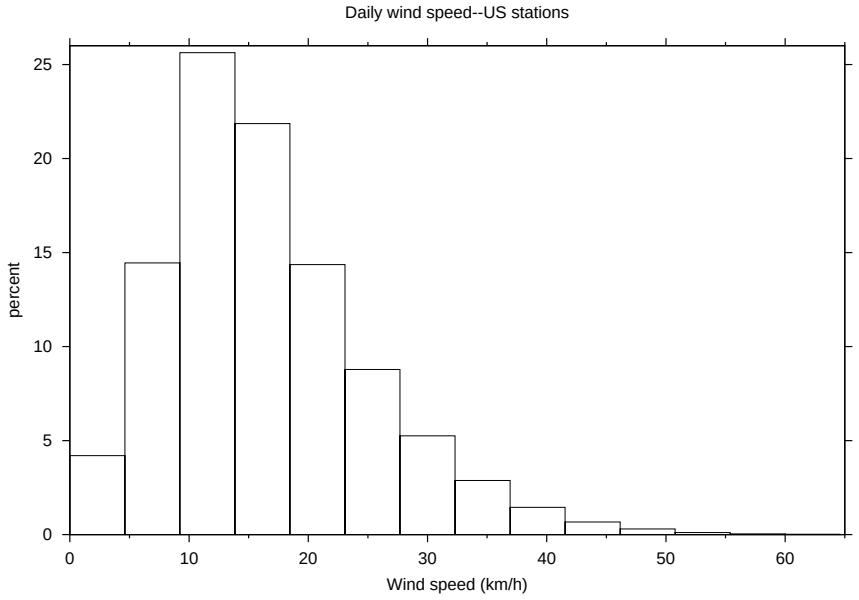

Figure 3.12: Histogram of U.S. subset of daily wind speed.

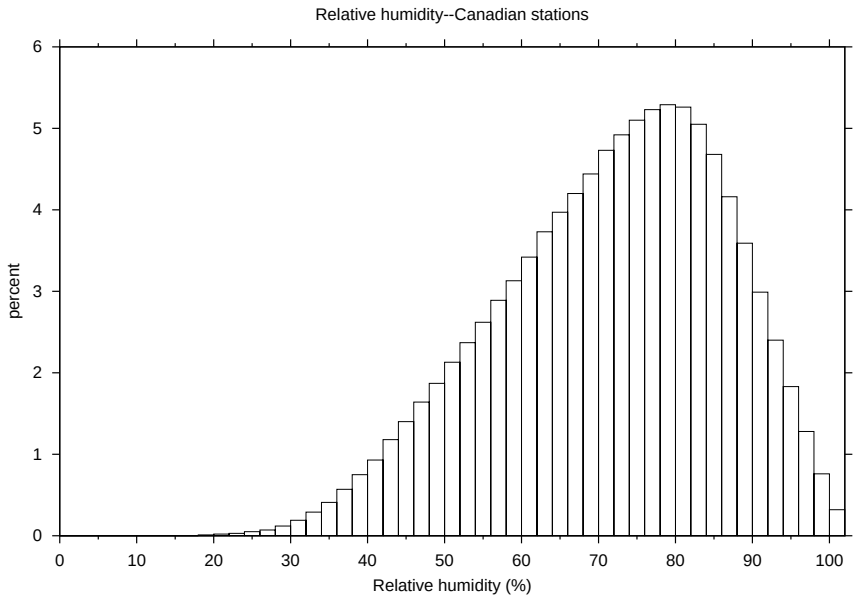

Figure 3.13: Histogram of Canadian subset of daily relative humidity.

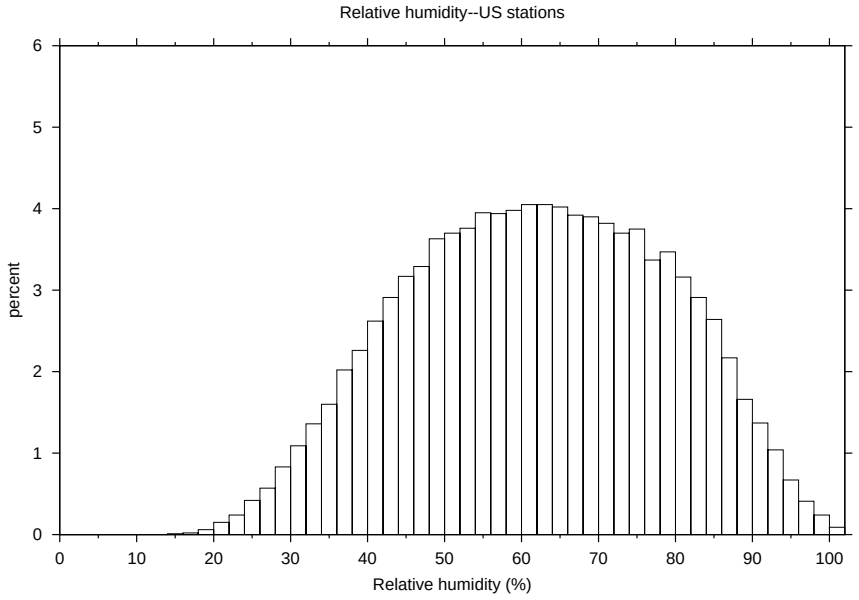

Figure 3.14: Histogram of U.S. subset of daily relative humidity.

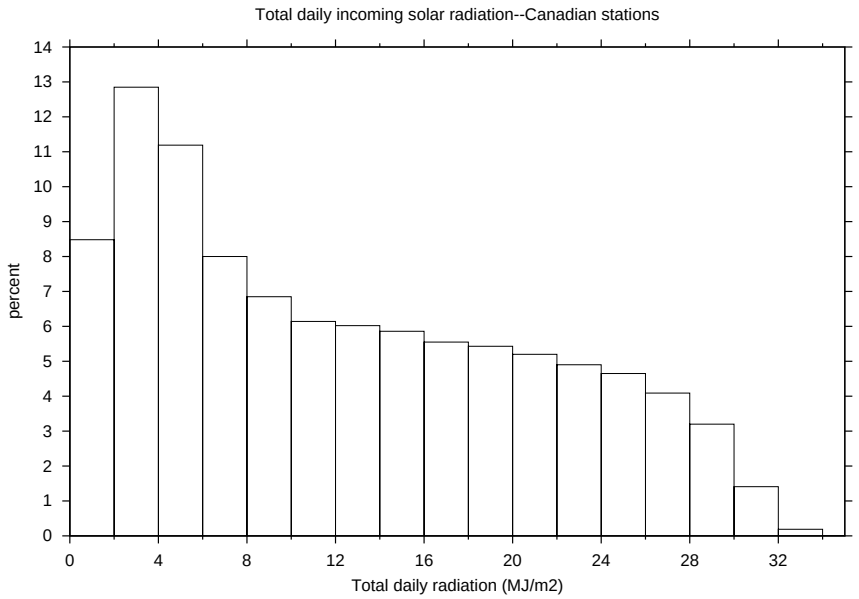

Figure 3.15: Histogram of Canadian subset of total daily incoming solar radiation.

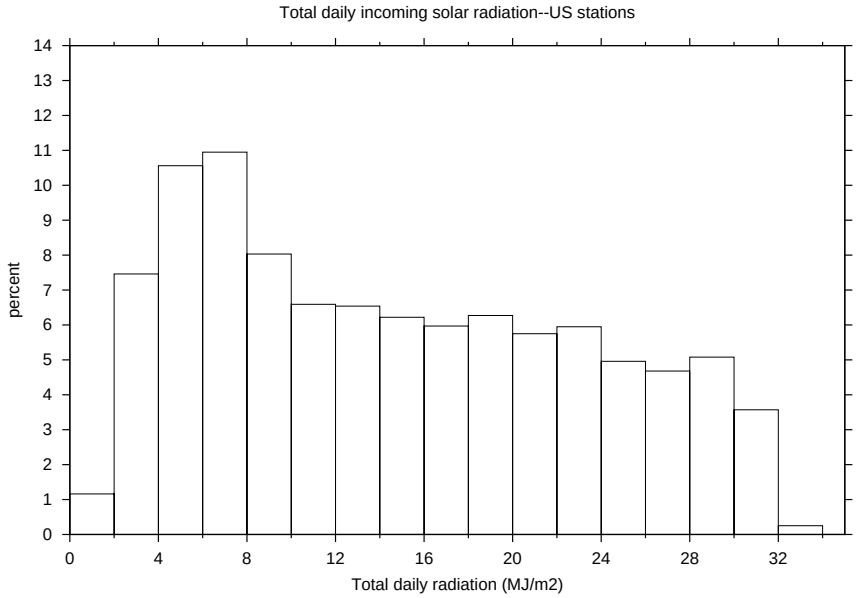

Figure 3.16: Histogram of U.S. subset of total daily incoming solar radiation.

For radiation (Figs. 3.15–3.16), the Canadian data give more values below 5 MJ/m^2 than the U.S. The trend falls off gradually as 30 MJ/m^2 is reached. Above 27 MJ/m^2, the U.S. radiation observations remain more or less constant to 32 MJ/m^2, whereas the Canadian data fall off rapidly.

The histogram differences between the U.S. and Canadian data shown in these figures are clear and indicate the existence of different climatic properties between the Canadian and U.S. regions under our investigation. The differences reflect both physical properties of the climate parameters and geographic locations of the regions. Thus the differences in histograms should not cause one to reject a data set from either country.

Temporal interpolation of missing dates

For a given climate parameter, it was not necessary that any particular station data file consist of a continuous sequence of days. It was only important that the combined data from all stations have this property for each parameter. Incoming solar radiation was the only parameter for which there were completely missing days of data, 29 days in total. Since one of the crucial project goals is to produce a complete data set with no missing days, this problem needed to be addressed in a scientifically appropriate way. For these 29 days scattered throughout the period of record, there are no station data as input on which to do a spatial interpolation; hence no output data on polygons can be computed without resorting to some form of temporal interpolation. This consists of computing a value for the missing day(s) based on the values of neighbouring days. One might think that it would be most appropriate to perform this step on the polygon values, which is the final output from the spatial interpolation. This would prevent any of the introduced "data" from affecting the spatial interpolation except where needed. Such a restriction is not necessary, however. For in the spatial interpolation, the calculation of each day is independent of the others. Hence if the station data of one day were somehow identical to another day, the two resulting

Table 3.3: Properties of the processed station data.

Parameter	Type	Precision & Units
Maximum Temperature	daily	0.1°C
Minimum Temperature	daily	0.1°C
Precipitation	daily total	0.1 mm
Relative Humidity	average of hourly data	1%
Radiation	daily total	0.1 MJ/m^2
Wind Speed	absolute average of hourly wind speed	0.1 km/h
Wind Direction	average of hourly wind vector	10° cw from true N, or the 8 or 16 points of the compass

spatial interpolations for these days would be identical. Therefore, a temporal interpolation occurring in the station data would give the same result as a temporal interpolation occurring in the final polygon data. This is provided, of course, that a temporal interpolation in the station data is used only when *no* stations are reporting data on that day. Thus we resort to a temporal interpolation on the data of two radiation stations in 29 instances to compensate for the lack of radiation data on those days. See Appendix A for a listing of the records which were introduced.

3.2 Processing the station data

Each climate parameter used in the interpolation has a particular unit of measurement with a given precision. Some parameters were derived from hourly data, representing either totals or averages. The particulars are given in Table 3.3. The radiation and wind data, in particular, presented unique processing challenges while preparing them for interpolation. The issues related to these parameters are presented below.

Radiation

The AES independently measures four radiation fields (RF) at every station which records radiation information. These four fields are (RF1) global solar radiation, (RF2) sky radiation (diffuse), (RF3) reflected solar radiation, and (RF4) net radiation. Because of the project's relation to agriculture, RF1 is studied here. It is defined as the total incoming direct and diffuse short-wave solar radiation received from the whole dome of the sky on a horizontal surface (Environment Canada, 1982a). The standard metric unit of radiation, which is used here, is the megajoule per square metre (MJ/m^2). AAFRD received the data in an hourly form and summed the entries into daily values for each station. The typical range for total daily incoming solar radiation is 0.0–35.0 MJ/m^2/day.

For the two U.S. radiation stations, global radiation (GRAD) was used. These were originally reported as hourly values in units of Watt-hours per square-metre (Wh/m^2), and were converted to MJ/m^2/day. Any days with missing hourly entries were not used because in all cases, the missing flags severely distorted the daily total when summed, and no attempt at interpolating the missing hours was considered.

Wind

Wind speed is the speed of air at a given point and is expressed in kilometres per hour (km/h). Wind direction is the direction from which the wind blows with respect to true north (Environment Canada, 1982b). Depending on the instruments used, wind speed is sometimes defined as the total wind run, which is the number of kilometres the anemometer cups have turned in one hour.

In computing the daily wind averages, two methods were employed. First, for the average wind speed, only the speed was used in the computation. This gives the average magnitude of the wind speed irrespective of the direction from which the wind is blowing. It is important to have this "magnitude-

only" estimate of the wind's behaviour when modelling soil erosion. For wind direction, however, it is more meaningful to gauge the average wind direction (of hourly values) by taking into account the wind speed at each hour of the day. A direction associated with a strong wind speed should carry more weight than a direction associated with a light wind speed. Thus one calculates the daily average wind direction by summing the individual components of the hourly wind vector (the north component and the east component). We call the daily average wind speed U and the daily average wind direction Θ_U. Let s_i be the horizontal wind speed, θ_i the horizontal wind direction for a given hour i, and N the number of hourly observations included in the average. Then the resulting wind speed and direction are,

$$U = \sqrt{U_e^2 + U_n^2} \tag{3.1}$$

and

$$\Theta_U = \arctan(U_e/U_n), \tag{3.2}$$

respectively, where

$$U_e = \frac{1}{N}\sum_{i=1}^{N} s_i \sin\theta_i, \quad U_n = \frac{1}{N}\sum_{i=1}^{N} s_i \cos\theta_i. \tag{3.3}$$

Note that in FORTRAN and other comparable programming languages, the function `atan2(x,y)`, which computes the arctangent of x/y, should be used in this context since it correctly places Θ_U in the proper quadrant based on the signs of U_e and U_n. The function `atan(x)`, where x is just a ratio, does not have this property.

Chapter 4

Implementation Details

The production of the final interpolated data sets for this thesis is a complex process, consisting of many stages, scripts, and FORTRAN programs. The variety and number of stages reflect the use of seven different climate parameters as input, in conjunction with the three output forms of EDP, SLC, and townships to arrive at the final product. One must consider using all stations within and without the province for all days under consideration, running different interpolation programs for each type of climate parameter, and running a final merge procedure to put all the pieces together according to the output specifications. This chapter describes the major steps in the process, gives an outline of the interpolation computing algorithm, and discusses various programming issues encountered in the production of the interpolated data.

4.1 Preparation of the station data as input

The daily climate data provided by AAFRD is in the form of a single text file for each observation station and is named according to the station's alphanumeric identifier.[1] See Section 3.1 on page 18 for a discussion of the quality

[1] A master station catalogue available from Environment Canada exists to look up information on each station such as latitude, longitude, duration of service, and

control procedures employed to ensure the quality of the station data before they are used in the interpolation.

Temperature and precipitation data are included together in the same station files since these parameters share the same stations, whose number is in the order of thousands. Wind, relative humidity, and radiation, however, have very few stations, on the order of hundreds and tens. The groupings are shown in Table 3.2 and are adhered to throughout the project. The raw station data files contain columns of real numbers with each line representing a separate daily record. Taking the temperature and precipitation data as an example, the first three columns are day, month, and year, while the next three columns are the daily maximum temperature, daily minimum temperature (in degrees Celsius), and daily total precipitation (in mm).

The station data must be processed into a form suitable for use by the interpolation engine. Due to the significance of 30-year periods for climate normals, such as 1961–90, the station data and the interpolation runs were originally partitioned into various time periods to reflect this time-based distinction. Once for each time period, the station files were examined for missing days and where found the necessary data fields were assigned the value `-999.9` to represent the missing records. This ensured that all station files had the same length and format, beginning on the same day and ending on the same day. This facilitated the use of a uniform data import mechanism adopted for the quality control tools and other programs which read station data files.

However, this strategy was abandoned upon consideration of the entire temperature and precipitation data set from 1901–2000. Forcing each station data file to include all 36,525 days of the century by filling in missing days

types of data collected. It can be obtained at `http://www.msc-smc.ec.gc.ca/climate/station_catalogue/index_e.cfm` or at `http://www.cmc.ec.gc.ca/climate/doc/station\_catalogue/station\_data\_catalogue.txt.gz`. See Section 4.3 on page 42 for details on the construction of the station information files particular to the work of this thesis from the master catalogue.

makes the storage and run-time memory requirements unreasonably large. For example, it would take (2611 stations $*$ 36,525 days $*$ (10 columns for the date + (3 variables $*$ 8 columns))) = 3.0 GB of `ASCII` data alone to store the temperature and precipitation station data, for which approximately 13% would be actual observations. Furthermore, it simplifies the programming and code management to place the data in one contiguous piece in order that records belonging to any particular day can be accessed as a group, and accessed randomly. This suggests the creation of a single indexed binary data file to accomplish such access.

4.2 The single binary data file

The "single binary data file" in the title above refers to the compilation of all the station data into four separate files, according to the groupings of the seven climate parameters. There is one file for temperature and precipitation, and another for wind, relative humidity, etc. This file is an indexed binary file. "Indexed" means that the data are sorted according to the date, and that the divisions between dates are recorded in an index file. The index file allows the data for any day to be looked up at random (also known as direct access) and is simply a listing of the record numbers for the first and last records of each day. "Binary" refers to the storage of the data in a binary, real (floating point) representation. In `ASCII` form, the temperature and precipitation files use 522 MB of storage space whereas in binary form they use 454 MB for the binary file plus an additional 1.3 MB for the index file. The improvement sought is not necessarily in storage size, but in (1) the decreased time required to read the data from the computer's hard disk, and (2) the freedom to read only one day of data at a time. The index and binary files are named `temppcpn.idx` and `temppcpn.bin`, respectively.

Now the stations are assigned an integer, called the station index, from $\{1, 2, \cdots,$ `NSTATIONS`$\}$ which is meant to replace the direct use of station IDs.

Table 4.1: Contents of a single record in the station data binary file. The variables in this record are stored as 8-byte real numbers, making the record length $8 * 4 = 32$ bytes. The station index is an integer (stored as real) from $\{1, 2, \cdots,$ `NSTATIONS`$\}$ which is linked to a table of station IDs.

	Variable
1	station index
2	maximum temperature
3	minimum temperature
4	precipitation

(The latter must be stored as character strings, and as such, are difficult to handle.) Each record in `temppcpn.bin` contains the four variables shown in Table 4.1, stored as 8-byte real numbers. Thus the record length for this file is $8 * 4 = 32$ bytes. Again, any missing parameters are filled in with `-999.9`, but if a station has no temperature or precipitation data for a given day, it is not included in the data file. Thus `temppcpn.bin` consists of blocks which contain all station records belonging to one day.

The opening of `temppcpn.bin` and the reading of `temppcpn.idx` is accomplished with the following FORTRAN code where `record_day` is an integer which is offset from the starting day of the project (`FIRSTDAY`), and `record_start` and `record_end` are record numbers.

```
c     binary station data file
      open(11,file='temppcpn.bin', access='direct',
     &        form='unformatted',recl=4*8)

c     index file for binary file containing
c     {day, record number of beginning, end}
      open(12,file='temppcpn.idx', form='formatted')
```

```
c     record_day(    1) is going to be equal to FIRSTDAY, and
c     record_day(NDAYS) is going to be equal to  LASTDAY

      do i=1,NDAYS
        read(12,25) record_day(i), record_start(i), record_end(i)
      enddo

      close(12)
  25  format(3I12)
```

To access the records for all stations on a particular day, the following code is placed within a loop which sets the value of the variable `day`.

```
c     initialize stationdata array
c     stations without data will be -999.9 because of this step
      do i=1,NSTATIONS
        stationdata(1,i)=-999.9          ! tmax
        stationdata(2,i)=-999.9          ! tmin
      enddo

c     determine where to start and finish reading for that day
      recstart = record_start(day)
      recend   = record_end(day)
      recdays  = recend - recstart + 1

      do i=1,recdays
        read(11,rec=recstart+i-1) a,b,c,d
        index=int(a)                     ! get the station index
        stationdata(1,index) = b         ! tmax
        stationdata(2,index) = c         ! tmin
      enddo
```

According to this scheme, which employs the binary file `temppcpn.bin` and its index file `temppcpn.idx`, the expansion of the station data set due

to the filling in of missing data (by station) only occurs in the interpolation engine at run-time. The expansion requires (2611 stations $*$ 2 variables $*$ 8 bytes) = 41 kB of storage. This occurs only one day at a time and therefore memory that is allocated to station data for that day gets reused. This is the particular advantage of this system and nothing prevents the user from directing the engine to work on a subset of the total number of days.

4.3 Station information files

The interpolation engine uses a key information file to keep track of the station data. There is one station information file for each of the four groupings of climate parameters. In particular, the engine needs to know the station identifier (station ID) and the latitude-longitude pair for each station. These are placed in a file named `stationinfo_1901.prn` (Table 4.2) which contains a single line for each station. The ordinal number of each line in this file corresponds to the station index mentioned in Section 4.2 above. Preserving this ordering of stations across the entire project is imperative because, in different contexts, the stations are referenced by this index instead of their station ID. The latitude-longitude pairs are represented by the integer format `ddmm dddmm` in the master catalogue where `d` and `m` represent degrees and minutes, respectively. To be usable, the pairs are converted into their decimal degree equivalents by the engine with the function `convert()` as follows.

```
real function convert(a)
integer a, degrees, minutes

degrees = int(real(a)/100.0)
minutes = a - int(real(a)/100.0)*100
convert = real(degrees) + real(minutes)/60.0
```

Table 4.2: An excerpt of the temperature and precipitation station information file, `stationinfo_1901.prn`. Station IDs are character strings. Latitude and longitude are integers of the format `ddmm dddmm`, which represents degrees and minutes.

stationID	latitude	longitude
1010066	4852	12317
1010235	4824	12329
1010595	4835	12331
1010774	4830	12321
1010780	4820	12338
1010960	4836	12328
1010961	4834	12327
1010965	4834	12326
1011467	4835	12325
10114F6	4834	12322
⋮		

The station information files are vital to the correctness of the interpolation, for they are responsible for placing the observed station data in their proper locations. A misrepresented latitude-longitude pair will result in misplaced observed data, which affects the grid point interpolations. In fact, the removal of a station's data from the interpolation process is accomplished by setting its latitude and longitude to $(0, 0)$—thus placing it far away from the province. For this reason, great care was taken in the construction of these files.

Each climate station is known by its identifier, the station ID. This alphanumeric string was used to look up the station's location and details of its observing program in the master climate station catalogue, during the construction of the station information files. A new ID is assigned to a station by AES whenever a major change occurs, such as a relocation of instruments or a change in observing program. This is indicated in the master catalogue by an entry under the new ID. A new station ID is not assigned for every change, however, so the master catalogue contains multiple entries under the same ID. In these cases, the station ID is listed for different time periods, sometimes with differing latitude-longitude pairs. This introduces the problem of deciding which latitude-longitude pair applies to the station ID for which we have data. Often, the difference in location is only a few minutes of latitude or longitude. But sometimes the locations differ by one or two degrees. A complete solution might attempt to use the multiple entries to split up the station's data into separate files for each of the locations given. This was not a feasible solution given the data in its present form and the number of cases for which this would be done (over 1,200). We decided to consider only the latitude-longitude pair of the first instance of the station ID in the master catalogue as a compromise. The effect this could have on nearest-station assignment or inverse-distance weighting would likely be small given the low density of stations.

4.4 Interpolation engine outline

This section discusses the specific steps taken by the interpolation engine during a single run of the program. See Appendix C for an abridged version of the wind interpolation code and Appendix D for temperature. As is typical coding practice for such applications, the run-specific parameters are defined at the top of the FORTRAN code and need to be set before each run. The parameters for unused scenarios are commented out so they can be easily recalled when needed. Such parameters include the type of interpolation to be executed (onto EDP polygons, SLC polygons, or township centroids), the number of stations (`NSTATIONS`), the number of polygons (`NPOLY`), the number of days (`NDAYS`), the number of grid points (`NPOINTS`), and the path where the station data files can be found (`STATIONPATH`). This makes the program flexible enough to handle different time periods and the various partitions of the province while keeping the code reasonably simple to use, understand, and maintain.

The actual nearest-station, inverse-distance, and hybrid steps are repeated for each grid point and repeated for each day in the temporal data space. This double-repetition is the major factor in determining the storage requirements and run-time speed for the entire program. For example, a typical SLC wind interpolation has the parameters shown in Table 4.3. Notice that a significant bottle-neck may occur in loops which run over all days and all grid points (that is, where the loop executes $10957 * 6746 = 73,915,922$ times). Therefore, one should minimize the number of times such loops are constructed—and they occur twice for nearest-station assignment. Furthermore, the code within these loops should be optimized for speed during the program design stage.

Program declarations. The parameters (`NSTATIONS`, `NPOLY`, `FIRSTDAY`, `LASTDAY`, `DATEOFFSET`, `NDAYS`, `NPOINTS`, and `STATIONPATH`, etc.) are specified here at the beginning of the program, which sets it to work

Table 4.3: Typical run-time parameters for SLC wind interpolation. Shown are parameters for two different time periods.

Parameter	1961–90	1901–2000
`NSTATIONS`	84	518
`NPOLY`	894	894
`NDAYS`	10,957	19,359
`NPOINTS`	6,746	6,746

on particular stations, polygon type, and time period. All variables and functions used in the code are also declared here.

Load the information files. In this stage, the program loads several information files. The key pieces of information are (1) the record numbers required to access the indexed binary station data file, (2) the station and grid point locations, (3) the list of grid points contained within each polygon, and (4) the names of the polygon output files to be produced. The station information files, in particular, were discussed in Section 4.3 above.

Calculate the distance table. The program proceeds to calculate and store the distance from every grid point to every station. The function `distance()` calculates the great circle distance between two points on a spherical Earth. Let the two points be located at (lat_1, lon_1) and (lat_2, lon_2). Then the great circle distance, d, between them is defined as

$$\begin{aligned} d = \arccos[&\sin(lat_1) * \sin(lat_2) \\ &+ \cos(lat_1) * \cos(lat_2) * \cos(lon_1 - lon_2)]. \end{aligned} \tag{4.1}$$

A mathematically equivalent formula, which is less subject to rounding

error for short distances, is

$$d = 2 * \arcsin \left[\sin^2 \left(\frac{lat_1 - lat_2}{2} \right) + \cos(lat_1) * \cos(lat_2) * \sin^2 \left(\frac{lon_1 - lon_2}{2} \right) \right]^{-1/2}. \quad (4.2)$$

The values returned by these functions are unit-less (in degrees of latitude) and must be multiplied by the factor 111.12 km/degree to convert the distance to kilometres. This is required to match the units of the temperature and precipitation length scales. Since the program needs to calculate the distance to each station for every grid point, a single nested loop calls the function `distance()` (`NPOINTS` $*$ `NSTATIONS`) times, which is only moderately expensive computationally.

Sort the distance table. To accomplish a nearest-station assignment or inverse-distance interpolation, the distances of stations with respect to a given grid point need to be ranked from nearest to farthest. The sorting of these distances employs a selection sort algorithm. This gives significant improvement over a simple bubble sort, considering that it works without resorting to a recursive algorithm (Kruse, 1987). This property keeps the sorting algorithm relatively easy to program and debug.

The sorting code loops (`NPOINTS` $*$ `NSTATIONS`) times and executes in a reasonable amount of time: under five minutes on an Athlon 900 MHz computer running FreeBSD 4.4. Under the present project, with the sorting of station distances required for at most 3,000 grid points, this amount of time is indeed reasonable. The execution time will become unmanageable, however, when one begins to increase the number of grid points, and more significantly, the number of stations. In reality, the inverse-distance interpolation probably needs only the first 20 stations closest to a grid point, and not the rank of all stations. By the 20th station, it is likely that the eight required stations with data would

have been found. But this cannot be known without first analyzing the day-to-day station density with respect to each grid point. Therefore, to keep things simple conceptually and programming-wise, the program blindly sorts all the distances to each station so that all station data is available for use. As the scaling effects become more of an issue for run-time in later investigations, one may wish to find a different way to handle this problem. As it stands, the sorting is done once, independent of the day or the data, based only on station and grid point locations. If one wants to take into account the actual availability of data on each day, then the sorting gets moved into a loop running over all days, which limits its feasibility as a solution.

Interpolation section. The final goal is the assigning of climate parameters to either polygons or townships. The latter are on the same dimensional scale as the 10 km-by-10 km grid, so a township's centroid is used as the interpolation point to determine the climate value for the township. Thus it is unnecessary to use grid points in this case. Instead, the township centroids are treated by the program in the same manner as for the regular grid points. The only difference is that the averaging-over-polygon step is omitted. Otherwise, the normal (polygon) operation of the program takes station data onto grid points and then onto polygons. A memory space limitation of the operating system prevented us from storing the grid point values of every day as intermediate calculations for use later on in the program (since (`NPOINTS` $*$ `NDAYS`) is very large).[2] Thus the station-to-grid point interpolation and the grid point-to-polygon averaging occur in succession under the same loop over all days.[3] Thus, if desired, outputting of the

[2]While it is possible to specify the maximum stack size and other such parameters when loading the operating system's kernel, a lack of experience in this area caused such tweaking to be fruitless. It should be possible, however, to do this successfully.

[3]In particular, this loop occurs in lines 216–302 for the wind interpolation code listed in Appendix C.

grid point interpolation (for plotting in GrADS, for example) would occur within this loop.

Since there is an averaging step which involves accumulating data over all grid points in the polygon, the program first initializes the polygon array to 0.0 before it can be used. It then enters into the over-arching day loop. Here the (`NDAYS` $*$ `NPOINTS`) loop occurs twice. First, the program loops over all grid points to find the nearest station(s) with data on that day. Then it loops over all grid points again to accumulate the grid point data for each polygon to which the grid point belongs. The number of grid points occurring in each polygon is recorded, and a final loop over all polygons turns the accumulation into an average by dividing through by this number.

Output stage. Here the final preparation of the polygon or township data is made. In the case of polygons, the output files are created anew and the data is rounded to the proper number of decimal places and written to the files. The output specifications are described in Section 4.7 below. For townships, the process is different because of their sheer number. Instead of creating new output files for each climate parameter and later merging the data together, only one set of 10 township output files exists (with each file spanning 10 years). These are updated each time an interpolation program is run by directly accessing and modifying each township record as necessary. Again, the data is formatted according to the output specifications. Both types of output require a single nested loop of (`NPOLY` $*$ `NDAYS`). The speed of executing the output steps is limited to a great extent by the writing and caching speed of the hardware and operating system.

After the execution of a single run of the interpolation program, the parameters are changed and the program run again. This facilitates the use of the same program for EDP polygons, SLC polygons, and townships, as

well as for different time periods. Similar programs are run to complete the interpolation for the remaining climate parameters. Then, in the case of polygons, several supporting shell scripts and FORTRAN programs are used to merge the data files together into the format specified in Section 4.7. See Appendix E for an example of a supporting script which is used to manipulate many files.

4.5 Optimization and memory access patterns

To produce source code which is optimized for speed, one must understand how the compiler turns code into machine instructions, and how these instructions are carried out by the computer. Since our interpolation algorithms require the storage and manipulation of large data sets, optimization with respect to memory access patterns becomes an important issue. The data are stored in very large, multidimensional arrays in their various forms: station data, grid point data, and polygon data. As these data structures are referenced, both for reading and writing, the best performance occurs when the coding is in harmony with the manner in which memory is accessed. The natural pattern is for the array indices to be increasing and unit sequential (Dowd and Severance, 1998). Because of memory caching—on-chip, in RAM, or in virtual memory—the computer grabs memory in blocks, regardless of the particular request. One can harness the power of this behaviour by requesting data in sequential pieces (or nearly so) to be covered by these blocks.

For a single dimensional array, the fastest access is obtained by stepping through the array one element at a time. For multidimensional arrays, the fastest access occurs when iterating over the array index which gives the smallest stride or step size in memory. This is due to the sequential storage of the "rows" and "columns" of a multidimensional array, namely how the rows and columns are arranged in memory by the compiler. In FORTRAN,

the index giving the smallest stride is the leftmost index; in C, it is the rightmost. A sample of a loop in FORTRAN which gives unit stride is shown below.

```
DO J=1,N
  DO I=1,N
    A(I,J) = B(I,J) + C(I,J) * D
  ENDDO
ENDDO
```

In C, it looks like this.

```
for (i=0; i<n; i++)
  for(j=0; j<n; j++)
    a[i][j] = b[i][j] + c[i][j] * d ;
```

So in FORTRAN, notice the innermost loop should be over the leftmost index. Depending on the conditions of the array size and which type of memory is being accessed, the difference between looping using a unit stride and stepping through an array using a larger-than-necessary stride can be quite significant. The implementation of unit stride memory access improved the run-time of one particular test-run of the SLC wind interpolation from 10 hours to 20 minutes! Therefore, it is important to be aware of this language-particular behaviour when designing code.

4.6 Months, leap years, and the use of Julian dates

Each item of data in this project has two important pieces of information associated with it. First is the physical location—that is, to which station, polygon, or township do the data belong. Second is the date. The 36,525 dates which span the project can give one quite a headache—and moreover,

Table 4.4: Julian day numbers of important dates.

Date	Julian day	Rank
31 Dec 1900	2,415,385	0
1 Jan 1901	2,415,386	1
1 Jan 1961	2,437,301	21,916
31 Dec 1990	2,448,257	32,872
31 Dec 2000	2,451,910	36,525

be a source of error—if they are not managed properly and in an efficient manner. The most difficult issue one encounters is the leap year. The 365.25-day orbital period of the Earth has forced modern society to adopt a calendar whose decades, years, and months have variable numbers of days in them. One must always be aware of whether the February of a given year has 28 days or 29 days. A simple way to eliminate this concern is found in the conversion of the date into some representation where each day is an integer and the next calendar day corresponds to the next integer in the sequence. Computers at their basic level work with dates in this manner, including both the operating system (Unix, for example) and applications (in particular, the C programming language). Since FORTRAN itself does not have built-in date functions, a well-known integer called the Julian day number is introduced to drive the interpolation code. Most frequently seen in astronomical applications, the Julian day is an assignment of a unique number to every day since 1 January 4713 B.C.E. Julian day 0 designates the 24 hours from noon UTC on 1 January 4713 B.C.E. to noon UTC on 2 January 4713 B.C.E. (Tøndering, 2001). Since the smallest time step in this project is the day, hours are ignored and with them, fractions of days. The climate data analysis begins with 1 January 1901 which corresponds to JD 2,415,386. The Julian day numbers of other dates relevant to the project are listed in Table 4.4.

The Julian day number is most useful because of its ease of computation. Conversion between calendar dates, namely {year, month, day}, and the Julian day is accomplished with two subroutines whose details need not concern the programmer who uses them (listed in Appendix B).[4] So the program runs over all days, setting the variable `day` to $\{1, 2, 3, \cdots, 36525\}$, which it can convert to a Julian day number by adding `2,415,385`. The {year, month, day} can then be computed by calling the subroutine `fromJD(jday, year, month, day)` which uses the parameter `jday` as input and parameters {`year, month, day`} as output. Consequently, if a particular section of code requires knowledge of the month or the year, then knowing the value of the variable `day` allows one to know both the month and the year. The precipitation frequency formula (2.4) on page 16 makes explicit use of the month, for example. Julian days were also used to check for duplicate and out-of-order station records during the quality control phase of the project. In particular, any listing of successive calendar dates forms a strictly increasing sequence of Julian days, which is easy to confirm by computer.

4.7 Output file specifications

The interpolated data sets consist of the seven climate parameters in three different geographical forms: EDP polygons, SLC polygons, and townships. For polygons, the output format is `ASCII` text files, with one file for each polygon, named according to the polygon ID. Thus there are 149 EDP files and 894 SLC files. Each file contains 36,525 lines, corresponding to the days in the period 1901–2000. Each line contains the following information (without the header).

[4] Several Julian day converters are available on the World Wide Web and were used to confirm the accuracy of these algorithms. In particular, see the converter by the U.S. Naval Observatory (2001).

```
YYYY MM DD  TMAX  TMIN  PCPN  RH   RAD    WS  WD
1961  1  1  -4.4 -13.7   0.0  85   2.9  13.3 199
```

This corresponds to FORTRAN format

```
(I4, 2I3, 3F6.1, I4, 2F6.1, I4).
```

`WS` represents wind speed and `WD` wind direction. Note that the date format `(YYYY MM DD)` orders the output file both alpha-numerically and chronologically. The format requires 50 characters per line (including the CR-LF characters to mark the end of the line). Therefore, the total file size is $36,525 * 50 = 1,826,250$ bytes, or 1.7 MB. Multiplying by the number of files gives 260 MB for EDP and 1.52 GB for SLC. The EDP data fits on one CD, but the SLC data is divided into three CDs according to SLC ID. Since the relative humidity, radiation, and wind data sets begin many years after 1 January 1901, the missing fields are assigned the value `␣-99.9` (which is six characters wide including the space). Despite taking up storage space, this ensures a uniform format throughout the files.

For township data, the climate parameters and time period are as above. However, the format of the data is binary integers (as FORTRAN and SAS understand them). This requires a multiplier to remove the decimal for certain fields. To keep the file size manageable, the data set is divided into 10 files of 10 years, namely 1901–1910, 1911–1921, $\cdots$, 1991–2000. The records are stored sequentially, sorted by date and then by township ID. See Table 4.6 for the record structure. Missing entries for relative humidity, radiation, and wind are assigned the integer value `-999`. Since there are 6,900 townships and the record size is 22 bytes, each 10-year file is $6900 * 22 * 3653 = 529$ MB in size. The number of days in a 10-year period is either 3,652 or 3,653 days, depending on whether it contains two or three leap years. The total size of the interpolated township data set is therefore 5.16 GB.

Access to a township record for a given date is as follows. First, choose the correct 10-year file corresponding to the date, and calculate the position

Table 4.5: Sizes of interpolated data sets.

Name	Storage type	Size
EDP	text (real)	260 MB
SLC	text (real)	1.52 GB
Township	binary (integer)	5.16 GB

Table 4.6: Record structure for township output file. Record length is 22 bytes. All entries are either 4-byte or 2-byte integers, including date and township ID. The date format is the single integer `YYYYMMDD`. Since relative humidity and wind direction are integers, they do not need a multiplier.

Variable	Byte Size	Multiplier
date	4	-
township ID	4	-
tmax	2	10
tmin	2	10
pcpn	2	10
rh	2	1
ws	2	10
wd	2	1
rad	2	10

of the date within the list $\{1, 2, \cdots, 3653\}$. This is the value for the variable `day`. Determine the township index, `township_idx` $= \{1, 2, \cdots,$ `NTOWNSHIPS`$\}$ by looking it up in the list of township IDs. The record number is then

```
irec = NTOWNSHIP*(day) + township_idx
```

and the corresponding read statement is

```
read(10,rec=irec) date,township_ID,
&                 tmax,tmin,pcpn,rh,ws,wd,rad.
```

With this method, random or sequential access of township records is possible. Therefore, one may query this database equally using sequences of days or sequences of township IDs. The key is the correct calculation of the record number. One can confirm that the correct record is read, since the date and township ID are stored in each record.

4.8 Implementation summary

The management of very large and complex data sets requires strategic planning and the consistent application of standards throughout a project. Specific challenges encountered during the interpolation of the 1901–2000 daily climate data included (1) pre-processing management of the 1.3 GB of station data, (2) run-time storage of the same, (3) optimization issues related to memory access patterns, (4) the proper and efficient handling of calendar dates, and (5) the organization and storage of the 7.0 GB of interpolated output data. The preceeding work was accomplished by applying a programming philosophy of flexibility, modularity, and consistent and precise documentation. Source code, information files, and data files were designed to be clear and easy to maintain, keeping them adaptable to changing project requirements.

Chapter 5

Results and Accuracy

5.1 Scatter plot method for comparing data sets

Preliminary results of the temperature and precipitation interpolations onto polygons were used to calculate 1961–90 normals and climate elements related to agriculture such as degree-days, heat units, growing season, and frost periods. These were plotted on EDP and SLC polygon maps similar to Figs. 1.1 and 1.2 as an initial step toward a new *Agroclimatic Atlas of Alberta* intended to extend the work of Dzikowski and Heywood (1990). With the new plots, a problem was noticed with `EDP828` and the other southern-most polygons in the province. In particular, this polygon was consistently colder than the surrounding ones on such plots as the annual total degree-days and the date of the first fall frost. Polygons along Alberta's southern border were noticeably colder as well. A comparison with the 1951–80 atlas, combined with an intimate knowledge of the agriculture of the area, suggested this part of the interpolation was in error. Data from the United States was not used to construct the old atlas, which indicated that the problem could originate with the U.S. data. Thus we performed a direct comparison of Canadian and U.S. station data along the border to investigate this hypothesis.

The method involved choosing a Canadian station and a U.S. station which lie across the border from each other on a similar line of longitude and comparing the climate data on each day. Both daily maximum temperature and minimum temperature were considered. Stations chosen covered at least 5,000 out of 10,957 possible days of data (1961–90) so that a substantial number of data points could be matched between the two stations. Two Canadian stations were also compared to each other as an experimental control. A scatter diagram was constructed for the climate parameter (maximum temperature, say) with the U.S. value on the x-coordinate and the Canadian value on the y-coordinate. If the two stations, which are close in proximity, make a measurement of the parameter, it is expected that the scattering of points should lie along the line $y = x$, indicating an agreement between the two measurements.

In the original production of the interpolated data set, the raw station data was transformed into files of equal length representing all 10,957 days of data—one file for each station, one line for each day. Missing data was represented by the value `-999.9`. Thus, it became an easy task to blindly match the two series of data. That is, days with missing values become points such as $(x, y) = (-999.9, -14.5)$ and consequently do not fall in the range of the plot, whereas available data do.

For the two Canadian stations `3031400` and `3044200`, which are approximately 219 km apart, the scattering of points lies along the line $y = x$ as expected (Fig. 5.1). For the U.S. and Canadian comparison, however, a very striking systematic error is visible. In Fig. 5.2, stations `3031400` (Canada) and `392` (U.S.) are merely 7 km apart and should be reporting very similar daily temperatures. But the cloud of points is distinctly above the line $y = x$, indicating that the U.S. data is colder than the Canadian data in a systematic way. The cloud of points follows a line with the same slope as $y = x$, but with a non-zero intercept. On the graph, one can imagine a vertical line drawn at 0°C. Somewhere along this line, in the middle of the cloud, is the

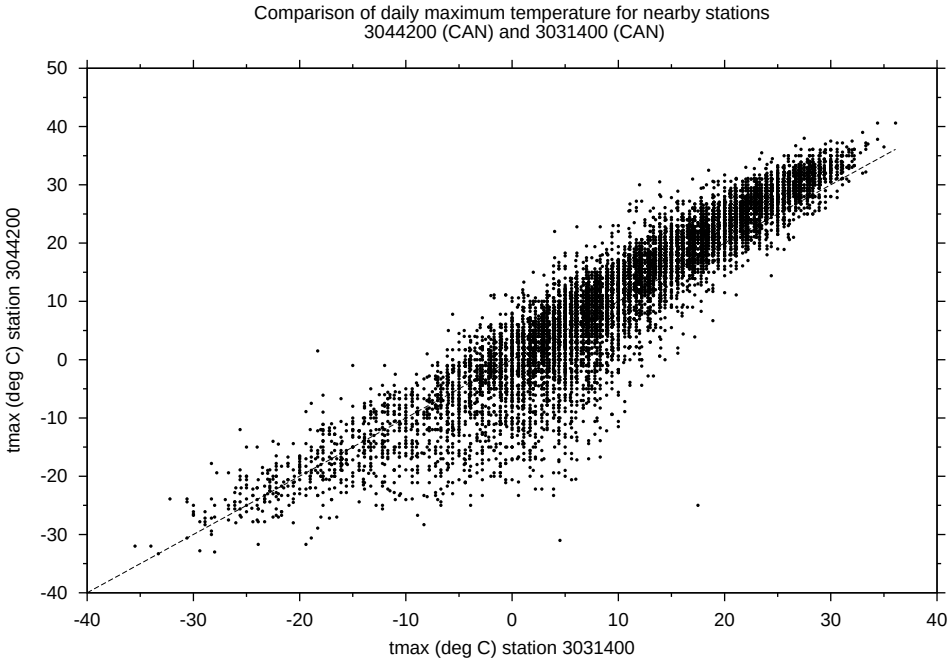

Figure 5.1: Comparison of daily maximum temperature (1961–90) for the two Canadian stations `3031400` and `3044200` which are approximately 219 km apart. The similarity of the cloud of points to the line $y = x$ indicates that both stations report similar values for the temperature field.

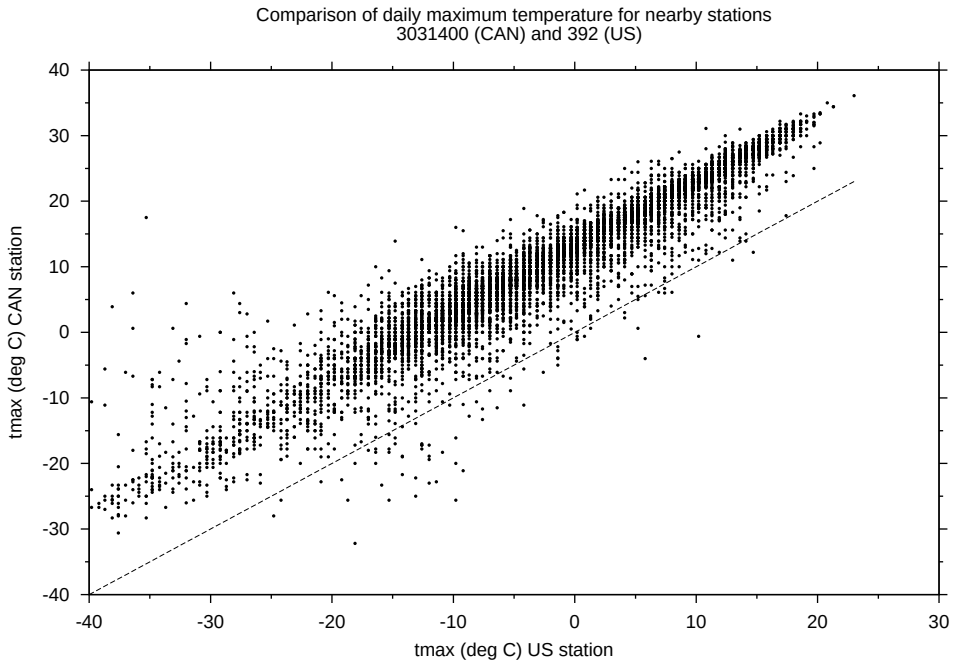

Figure 5.2: Comparison of daily maximum temperature (1961–90) for the Canadian station `3031400` and the U.S. station `392` which are approximately 7 km apart. The displacement of the cloud of points from the line $y = x$ indicates that the U.S. station is reporting significantly lower values for the temperature field than the Canadian station. A regression analysis estimates the offset from the line to be $(12.62 \pm 0.03)^\circ$C.

y-intercept, which can be estimated to be between 5°C and 15°C. This means that the U.S. daily maximum temperature data could be at least 5°C too cold and possibly as bad as 15°C. A regression of the (x, y) comparison formally estimates the intercept at (12.62 ± 0.03)°C with a correlation coefficient, R, of 0.957.

Further comparison of other station pairings shows the same discrepancy between U.S. and Canadian station data for daily maximum and minimum temperature. One hypothesis is that the conversion between Fahrenheit to Celsius was programmed incorrectly, resulting in the 13°C difference. Regardless of the origin of the problem, this peculiarity with the station data has since been investigated and eliminated with the new 1901–2000 U.S. data set.

The same technique described above was applied to daily precipitation data, but the more localized nature of the precipitation field did not lend itself to this method. In particular, the cloud of points did not form around the line $y = x$ but filled the corner of the first quadrant near the origin $(0, 0)$. In other words, a record of precipitation at one station did not correspond well to that of another station, regardless of the proximity of the two stations. But for temperature, and perhaps other climate fields, the scatter-plot method serves as a useful tool for comparing different estimates and/or samples of these fields. For example, this method was used to compare the interpolated wind speed and direction for a given polygon with nearby station data. Small polygons whose grid points only inherited values from a single nearby station would form a scattering of points with this station that lay perfectly on the line $y = x$. Larger polygons that would inherit data from several nearby stations would result in scatter diagrams showing a spreading out from $y = x$. Thus the method serves as a useful, localized check of the correctness of the interpolation algorithms.

5.2 Station density

Ratio of observation

There are several ways to evaluate the quality of the interpolated data. One of these is to have some concept of the density of observations which are used as input to the interpolation. In general, the accuracy of interpolation is inversely proportional to the density of stations. If there are a large number of stations in a region, the interpolation error should be small. However, it is not straight forward to define a parameter which reflects the station density, particularly when the number of stations changes with respect to time.

An indicator of station density over time, adopted in this thesis, is the ratio of observation, or the percent-potential of data, R. It is defined as,

$$R = \frac{\text{\# of daily data records for all stations}}{(\text{\# of stations}) * (\text{\# of days})} * 100\%, \tag{5.1}$$

and indicates how complete the station records are. That is, a ratio of observation of 100% would mean that all station records for a parameter have an observation for all days in the experiment. Table 5.1 gives the ratio of observation for the climate parameters over their respective time periods. The higher ratios reflect the shorter periods-of-record for the parameters wind, relative humidity, and radiation. A more interesting picture is given by calculating R by year. The number of potential stations is held constant throughout the century so the yearly ratios show how the actual number of observations compared to the potential number of observations changes with time. The yearly ratios of observation for temperature and precipitation are shown in Figure 5.3. The figure indicates that the temporal density of stations is not constant and that the highest temporal concentration of data occurs in the period 1965–90. Notice the sharp decrease over the period 1990–2000. While this may reflect an actual decrease in the number of stations in operation, it is more likely due to the lag produced by the time required for integration of new data into the AES climate data archive.

Table 5.1: Ratio of observation for each climate parameter.

Parameter	# of stations	# of days	R (%)
Temperature	2,611	36,525	12.0
Precipitation	2,611	36,525	15.2
Wind	518	19,359	29.7
RH	263	15,890	37.0
Radiation	18	19,359	35.9

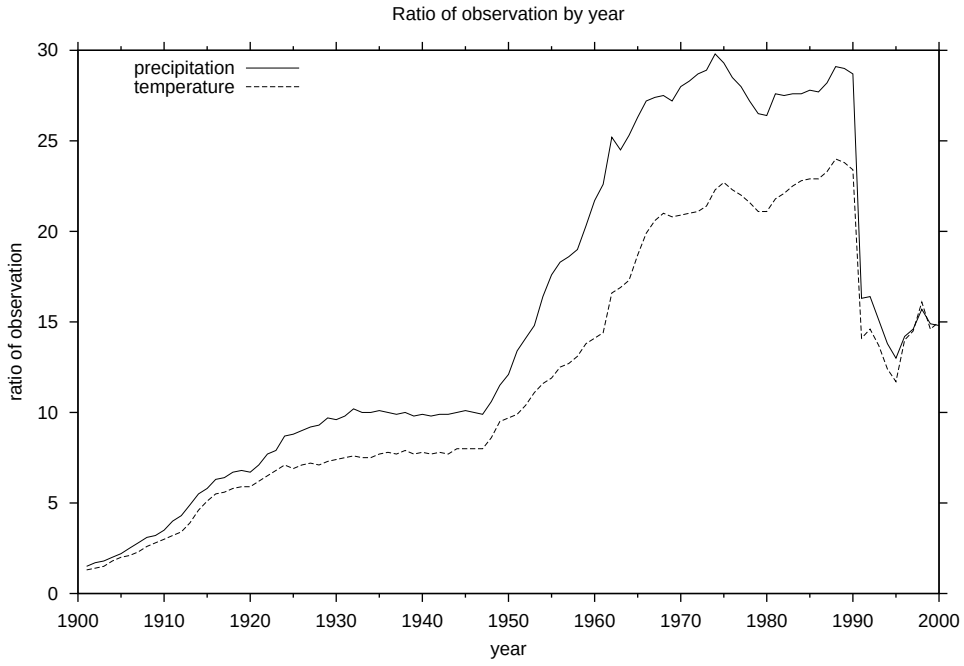

Figure 5.3: Ratio of observation for temperature and precipitation stations by year. This indicates how the number of observations varies with time.

Station counts

The first measure of station density given earlier in this section, the ratio of observation, looks at a bird's eye-view of the data set, without considering the mechanics of the interpolation method. An alternative approach is to look at the density of stations "up close," that is, at each grid point as the interpolation takes place. One can assign a measure of the station density, called the station density index, to each grid point for each day to accompany the interpolated climate parameter. A preliminary step in exploring this idea is to count the number of stations used at each grid point. This applies only to inverse-distance weighting (temperature) and the hybrid method (precipitation) since they both use a variable number of stations, depending on available data. The station count will be tallied for all grid points, over all days, applying to the province as a whole. A proposed station index, which is assigned to each grid point individually, will be described later in the section.

The method of counting is simple. As the interpolation runs over all days and all grid points, the program keeps track of the number of stations used in the interpolation.[1] Figure 5.4 shows the behaviour of the grid point interpolation for temperature (1961–90), which uses inverse-distance weighting. According to our particular algorithm design, the interpolation uses no more than eight of the nearest stations to the grid point. As seen in the figure, this occurs 83.0% of the time. Recall the rule that states if there are fewer than eight stations within the temperature length scale (200 km), then only these stations are used. This is represented by station counts 1–7. In the case of finding only one station (a station count of 1), the inverse-distance formula degenerates to a nearest-station assignment. If no stations are found within the length scale of the grid point, then the inverse-distance algorithm seeks the nearest station outside of the length scale. This is represented by a station count of 0. Thus, 0 and 1 are nearest-station assignment, but for 0,

[1]This step is shown explicitly on line 292 of the temperature interpolation code given in Appendix D.

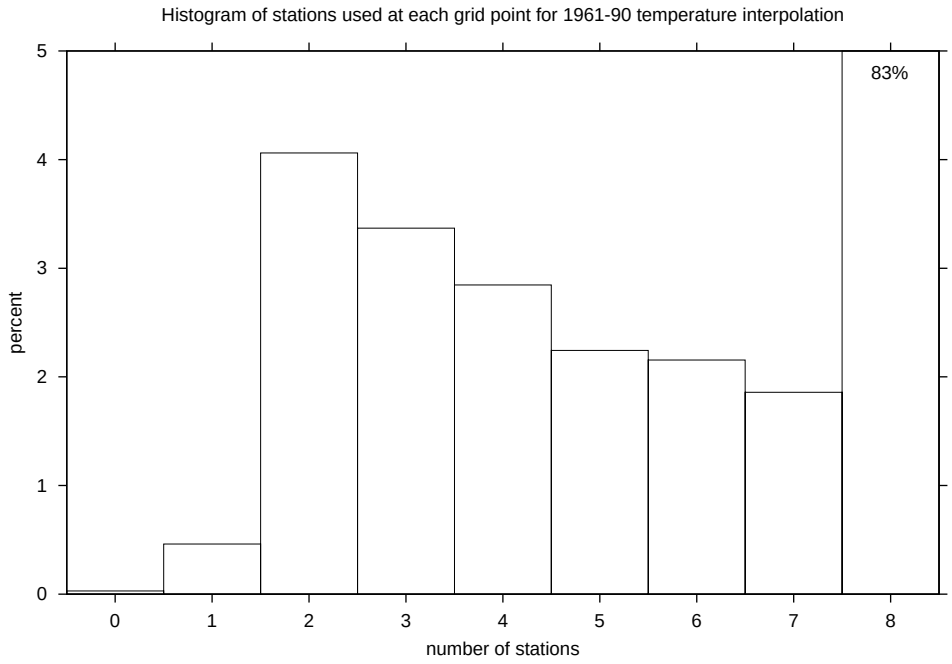

Figure 5.4: Histogram of the number of stations used at each grid point for the 1961–90 temperature interpolation.

the station is outside of the 200 km radius while for 1, it is inside. The total experimental space is the number of grid points multiplied by the number of days for the interpolation: $6633 * 10957 = 72{,}677{,}781$ in this case.

The figure shows that the station density is quite high overall during the period 1961–90. The full count of eight stations occurs for 83.0% of all grid point-days. This means the remaining 17.0% of the grid point interpolations derive their climate value from less than eight stations. Very seldom did the case of nearest-neighbour assignment (0 or 1) occur (less than half a percent). Therefore, the histogram of station counts shows that the inverse-distance algorithm had sufficient data during the period 1961–90 given the choice of length parameter and limit on station number.

The situation is almost as good for the temperature interpolation of the entire 1901–2000 period, shown in Fig. 5.5. There is a noticeable increase

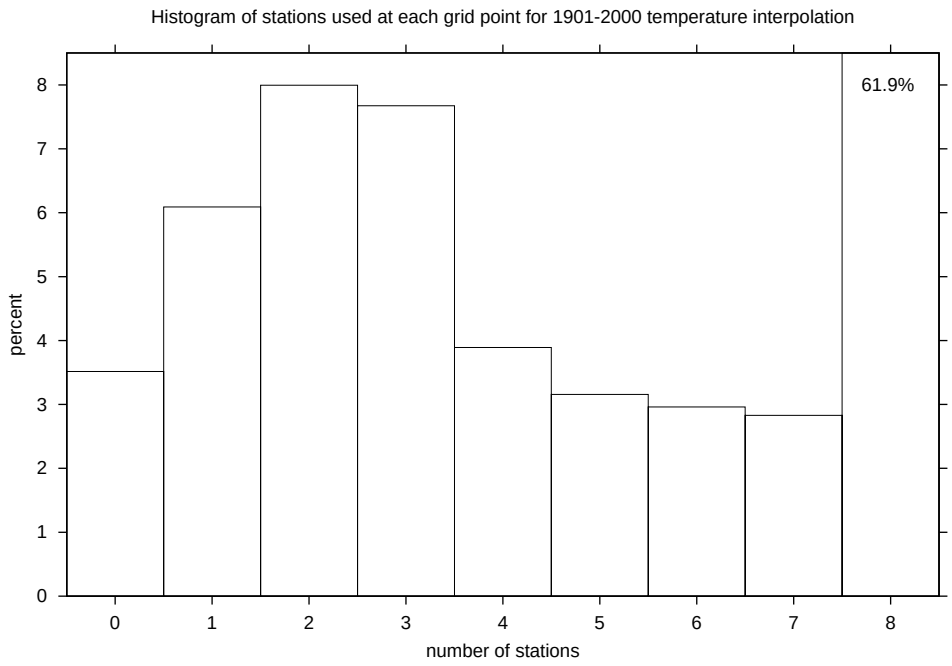

Figure 5.5: Histogram of the number of stations used at each grid point for the 1901–2000 temperature interpolation.

in station counts of 1–3 stations. This implies that less data is available, preventing the use of all eight nearest stations. The algorithm resorted to nearest-neighbour assignment more often, about 9.5% as shown by counts 0–1.

The 1901–2000 interpolation for precipitation is much worse, however. Recall that for precipitation, the hybrid method is used. This requires an inverse-distance interpolation to obtain daily precipitation values on grid points. It is at this stage that the stations used are counted. The results are shown in Fig. 5.6. In this case, only 10% of the time are eight stations available for the interpolation. The algorithm resorts to a nearest-neighbour assignment within the length scale 15% of the time, and without the length scale 37% of the time. This dramatic shift in station counts is due to the shorter length scale used for precipitation (i.e. 60 km) reflecting its more

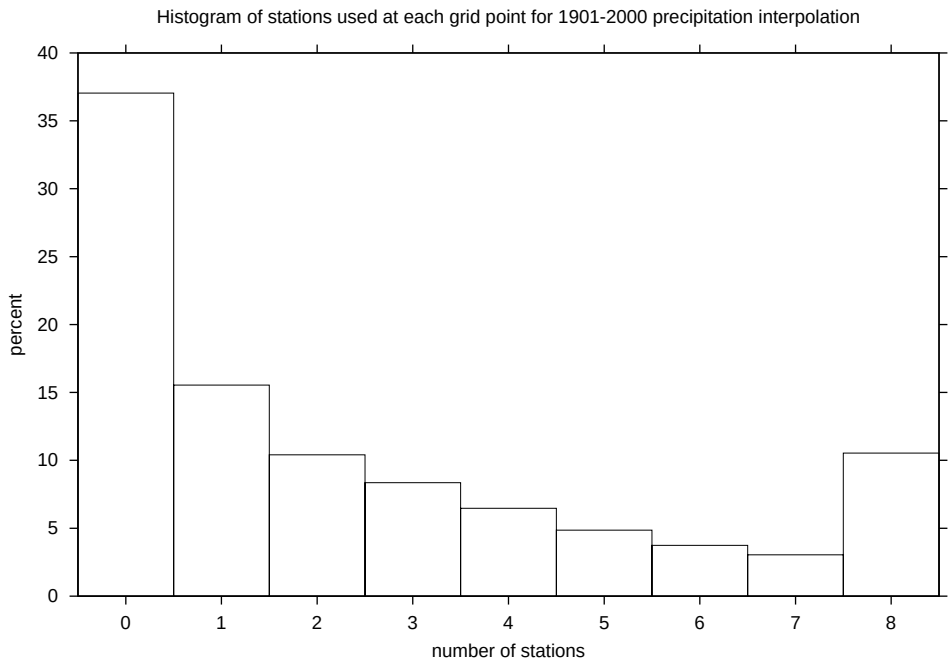

Figure 5.6: Histogram of the number of stations used at each grid point for the 1901–2000 precipitation interpolation.

localized field structure. It would be of interest to know the distance to the nearest stations referenced in those cases. Based on the previous figure for temperature (Fig. 5.5) we know that only 4% of the time the distance is greater than 200 km, since stations recording precipitation also record temperature, in general. Therefore, the counting of stations is a good method to examine the effects of different length scales. It can also can be used to explore the quality of the interpolated data over various time periods.

A proposed station density index

To have a measure of station density which can be displayed spatially on a map, one must calculate such an index at every grid point. For a nearest-station assignment, the most logical measure is the actual distance to the

one station providing the climate value. Thus, if looking at a map of Alberta showing this particular station density index for a given time period, one would see the areas of the province containing grid points that obtained their data from nearby and areas whose data come from more distant stations. This map should give a similar picture to a Thiessen polygon partitioning of the province, for Thiessen polygons show the region of nearest-neighbour influence for each station.

For inverse-distance weighting, the station density index should also be related to the distances of stations included in the interpolation. Since our inverse-distance scheme has the behaviour of using at most eight stations within a specified length scale, which then resorts to a nearest-station assignment if none are found within this distance, then an appropriate station density index should consider the distances to the eight nearest stations, irrespective of the length scale. Therefore, if the algorithm resorts to a nearest-station assignment in a given case, then the notion of distance built into the station density index will reflect the sparsity of stations. The station density index, ρ_j, at grid point $\hat{\mathbf{g}}_j$, appropriate for the interpolations which make use of the inverse-distance algorithm (temperature and precipitation) is suggested to be the weighting factor in the inverse-distance equation (2.1), namely,

$$\rho_j = \left(\sum_{i=1}^{M_j} \frac{1}{d_{ij}} \right)^{-1} . \tag{5.2}$$

The above is a proposed method for calculating a station density index at each grid point used in the interpolation. Experiments need to be conducted to test its usefulness as an indicator for station density.

5.3 Cross-validation of interpolation methods

The most effective method now commonly used to assess the error of climate data estimation is cross-validation (Cressie, 1993). The procedure compares estimated data for a point to observed station data at that point. Of course

the station data are withheld from the estimation; the data from other stations are interpolated to the station location. The statistics for the difference, or errors, between the true data and the interpolated data are used to evaluate the accuracy of the interpolation scheme.

To evaluate the accuracy, three types of errors were computed:

(i) Root mean square error (RMSE)

$$RMSE = \left[\frac{1}{K}\sum_{t=1}^{K}\left(X_{true}(t) - X_{estimate}(t)\right)^2\right]^{1/2}, \tag{5.3}$$

where K is the number of days used for cross-validation studies, t is time in days, and X denotes a climate parameter at a cross-validation location.

(ii) Mean absolute error (MAE)

$$MAE = \frac{1}{K}\sum_{t=1}^{K}|X_{true}(t) - X_{estimate}(t)|. \tag{5.4}$$

(iii) Mean biased error (MBE)

$$MBE = \frac{1}{K}\sum_{t=1}^{K}\left(X_{true}(t) - X_{estimate}(t)\right). \tag{5.5}$$

Since the polygon values are obtained from grid point values, the cross-validation is performed for both grid points and polygons. For grid point cross-validation, five long term stations distributed from south to north are considered. They are, in increasing order of latitude, Lethbridge CDA 3033890 (49°42′ N, 112°47′ W), Lacombe CDA 3023720 (52°28′ N, 113°45′ W), Edmonton Intl A 3012205 (53°18′ N, 113°35′ W), Beaverlodge CDA 3070560 (55°12′ N, 119°24′ W), and High Level A 3073146 (58°37′ N, 117°10′ W). The total number of days for cross-validation is 13,514 covering the 37-year period 1961–97. Hence, the total number of data entries for cross-validation is 67,570 minus the days without data at the cross-validation stations. Since

Table 5.2: Errors assessed by cross-validation for Edmonton station 3012205, (53°18′ N, 113°35′ W).

		Tmax (°C)	Tmin (°C)	Pcpn (mm)
Inverse-distance method	RMSE	1.75	2.16	2.36
	MAE	1.10	1.57	0.81
	MBE	-0.07	-0.67	-0.03
Nearest-station method	RMSE	2.10	2.83	3.30
	MAE	1.40	2.10	1.08
	MBE	-0.15	-0.79	0.02

day-to-day temperature and precipitation anomalies are normally independent of each other, the data for each day may be considered as an independent sample.

The RMSE, MAE, and MBE results for Edmonton, Lacombe, Lethbridge, and Beaverlodge are comparable. The magnitude of the errors for the Edmonton station are shown in Table 5.2.

For the inverse-distance method, the RMSE for Tmax ranges from 1.37–3.19°C, Tmin from 1.79–3.22°C, and Pcpn from 1.75–2.84 mm. For the nearest-station assignment method, the RMSE for Tmax ranges from 1.97–2.91°C, Tmin from 2.48–3.72°C, and Pcpn from 2.39–3.30 mm.

The errors are small for Lacombe, Edmonton and Beaverlodge stations, since these areas are flat and have higher station density. The station density in the Lethbridge area is also higher, but the topographic influence makes the errors slightly higher than Lacombe and Edmonton. The errors are larger for the northern-most station, High Level. The station density in this area is much lower. The mean station distance, defined by the sum of the mutual distances between any two points divided by the total number of distances, is about 105 km. Also the data stream of this station is short, less than 10 years compared to others of 37 years. Thus, the cross-validation errors

for this station are not considered representative, and the large errors of the nearest-station-assignment method for this station are not included in the error summary of the above paragraph.

The RMSE, MAE, and MBE errors above are considered measures of the goodness of fit to mean conditions. Our computational results show that the error for Tmax is usually smaller than that for Tmin. The nearest-station assignment method usually produces RMSE and MAE errors around 20–30% larger than the inverse-distance method. This result is expected, since the inverse-distance method yields a smooth field and the true daily weather distributes randomly on the positive or negative side of the smooth field. Thus, the field generated by the inverse-distance method has a smaller variance than the true field.

Three types of errors, RMSE, MAE, and MBE, were calculated for the five cross-validation stations. The results for the Edmonton station are listed in Table 5.2. The errors indicated that the inverse-distance method appears to generate more accurate results. The inverse-distance method over-smoothed the interpolated fields, particularly the precipitation field. We computed the sample variances for the data of the five cross-validation stations, and the variance results are shown in Table 5.3. Here, the variances are computed from the daily anomaly data. For each day in a year, the 1961–90 climatology is computed. The anomalies for a station are with respect to this climatology. The variance of the station is computed according to this anomaly data for 1961–90. Table 5.3 indicates that the inverse-distance method reduces the sample variance. For temperature, the reduction is small. The average of the five stations is less than 5%. For precipitation, the reduction is over 10%. (The results from the High Level A station were not representative—and hence excluded—because of the station's short record, namely 1,072 days during the 30-year, cross-validation period, 1961–90.)

The variance for the precipitation resulting from the nearest-station method is almost the same as that of the observed data. Thus, considering

Table 5.3: Sample variances of data from five cross-validation stations. Units are $^\circ C^2$ for Tmax and Tmin and mm^2 for Pcpn.

Station Name	Observed Data			Inverse-Distance			Nearest-Station		
	Tmax	Tmin	Pcpn	Tmax	Tmin	Pcpn	Tmax	Tmin	Pcpn
Edmonton Intl A	7.14	6.37	3.96	7.06	6.17	3.43	7.08	6.48	3.94
Lacombe CDA	7.24	6.10	3.76	7.20	5.91	3.53	7.42	6.22	4.05
Lethbridge CDA	7.65	6.71	3.79	7.06	6.17	3.29	7.70	6.49	3.79
Beaverlodge CDA	7.45	6.68	3.97	7.35	6.85	3.54	7.84	7.61	4.05
High Level A	7.08	6.77	3.31	7.16	6.74	3.82	7.70	7.35	3.57

the need to preserve the second moment, i.e. variance, the nearest-station method was selected as the preferred interpolation method. This is the reason the precipitation frequency was computed by the hybrid formula (2.4) in Section 2.3.

Let us consider the number of days with precipitation per month for the five cross-validation stations. For each cross-validation, three data sets exist: the observed data at the station, the interpolated data from the inverse-distance method, and the interpolated data from the nearest-station method. The cross-validation results for the Lacombe station `3023720` are shown in Table 5.4. The number of days with precipitation from the observed data and the number from the nearest-station interpolation are about the same, while the number from the inverse-distance interpolation is too large by about 50–100%. Such a big percentage is unexpected, although it is not surprising that the inverse-distance method yields too many precipitation days. Cross-validation results from other stations support the same conclusion.

We also validated the hybrid method on the five EDP and five SLC polygons in which the five cross-validation stations are located. The inverse-distance method yielded a result of daily precipitation and the formula (2.4) revised the result. The revised result had a larger variance than the one generated by the inverse-distance method. Table 5.5 shows the variance of the precipitation for the five polygons, before and after the revision. The

Table 5.4: Number of days with precipitation per month at Lacombe station 3023720 (52°28′ N, 113°45′ W) .

Month	Observed	Inverse-distance	Nearest-station
1	9.30	16.13	8.92
2	7.13	12.33	7.07
3	6.83	13.90	6.69
4	6.87	12.80	5.68
5	10.33	16.87	10.19
6	13.53	20.43	13.92
7	14.20	21.23	14.22
8	12.47	18.97	12.61
9	11.00	16.73	11.31
10	5.80	12.00	5.31
11	7.03	12.97	6.33
12	7.60	14.50	7.62

Table 5.5: Precipitation variances of the inverse-distance results and the revised results over the cross-validation polygons (units are $(\text{mm})^2$).

Polygon	Revised	Inverse	Polygon	Revised	Inverse
EDP727	3.82	3.25	SLC433	4.04	3.41
EDP737	4.10	3.41	SLC518	3.98	3.50
EDP793	3.65	3.18	SLC644	3.67	3.22
EDP598	4.09	3.49	SLC15	4.06	3.48
EDP586	3.63	2.82	SLC723	3.63	3.24

variance of the revised precipitation is about 10–20% higher than that of the inverse-distance results, which is the size of increase we intended to achieve. The revision does not change the monthly total precipitation, but it changes the temporal distribution and hence the amount of daily precipitation. The revised precipitation overcomes the problem of the over-smoothed inverse-distance results, which have too many precipitation days and too little precipitation each day. Table 5.6 shows the precipitation days per month in two of the ten cross-validation polygons: SLC518 and EDP793. The precipitation frequency results produced by the hybrid method for polygons are comparable to those for cross-validation stations (Table 5.4).

The hybrid method can also preserve the spatial localization of precipitation, while the inverse-distance method and other smoothing methods spread precipitation domains. The localization is particularly important in summer and significant for the climate input of soil quality models, since water erosion of soil is mainly due to extreme storm events. Figs. 5.7 and 5.8 show fields of a major storm and small, scattered storms interpolated by the hybrid method.

The remaining cross-validation question is the goodness of fit for polygons with respect to mean conditions. Since the true value of a polygon average can never be measured, cross-validation experiments cannot be used

Table 5.6: Number of precipitation days on two polygons computed from inverse-distance and hybrid methods.

	SLC518		EDP793	
Month	Inverse-distance	Hybrid method	Inverse-distance	Hybrid method
1	18.03	8.60	19.03	7.73
2	14.03	6.83	15.43	5.73
3	15.37	6.17	18.53	7.40
4	14.23	5.93	18.77	6.93
5	18.57	9.33	20.43	9.10
6	21.93	12.67	21.27	9.43
7	23.10	13.40	19.90	7.37
8	21.00	11.70	19.13	7.43
9	18.10	9.83	16.63	7.07
10	13.20	4.87	14.03	4.63
11	14.57	6.57	14.97	5.47
12	16.63	7.20	18.70	7.83

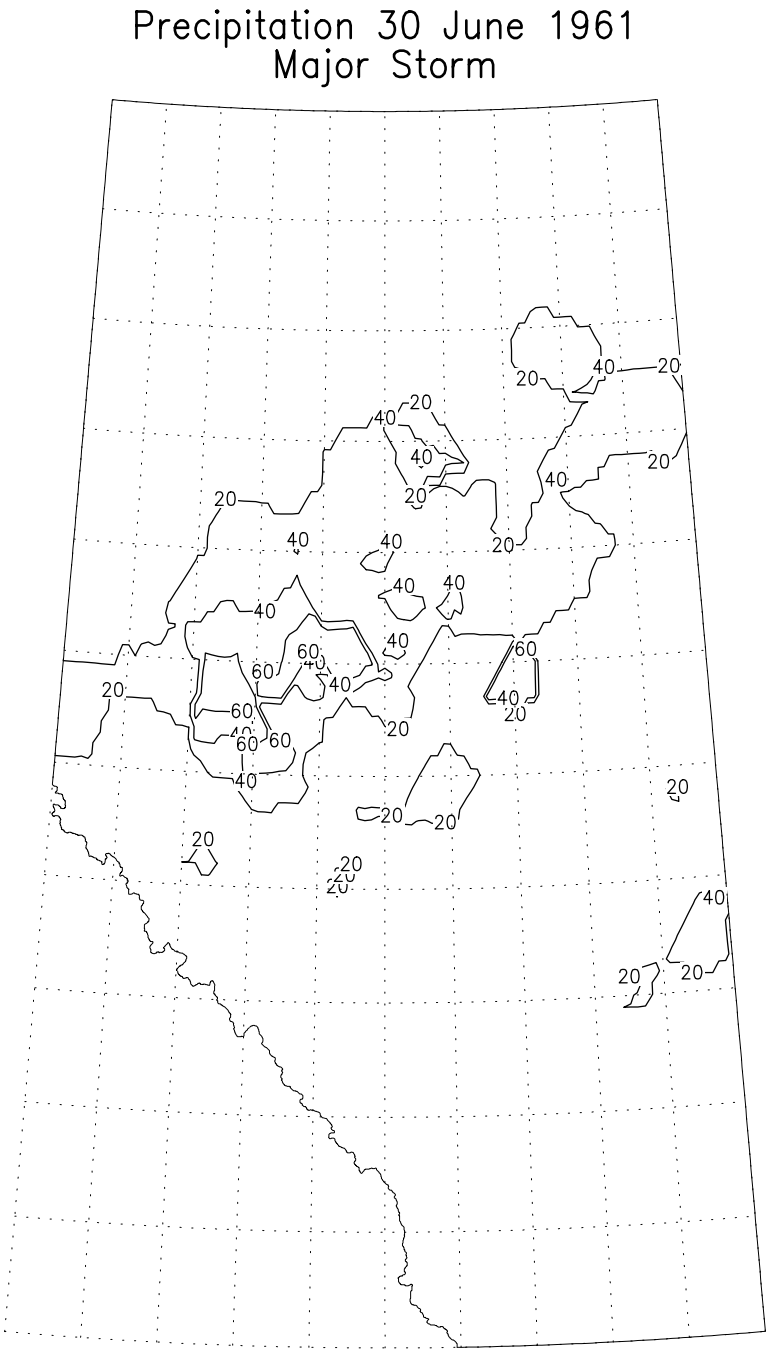

Figure 5.7: Precipitation as interpolated by the hybrid method for a major storm on 30 June 1961.

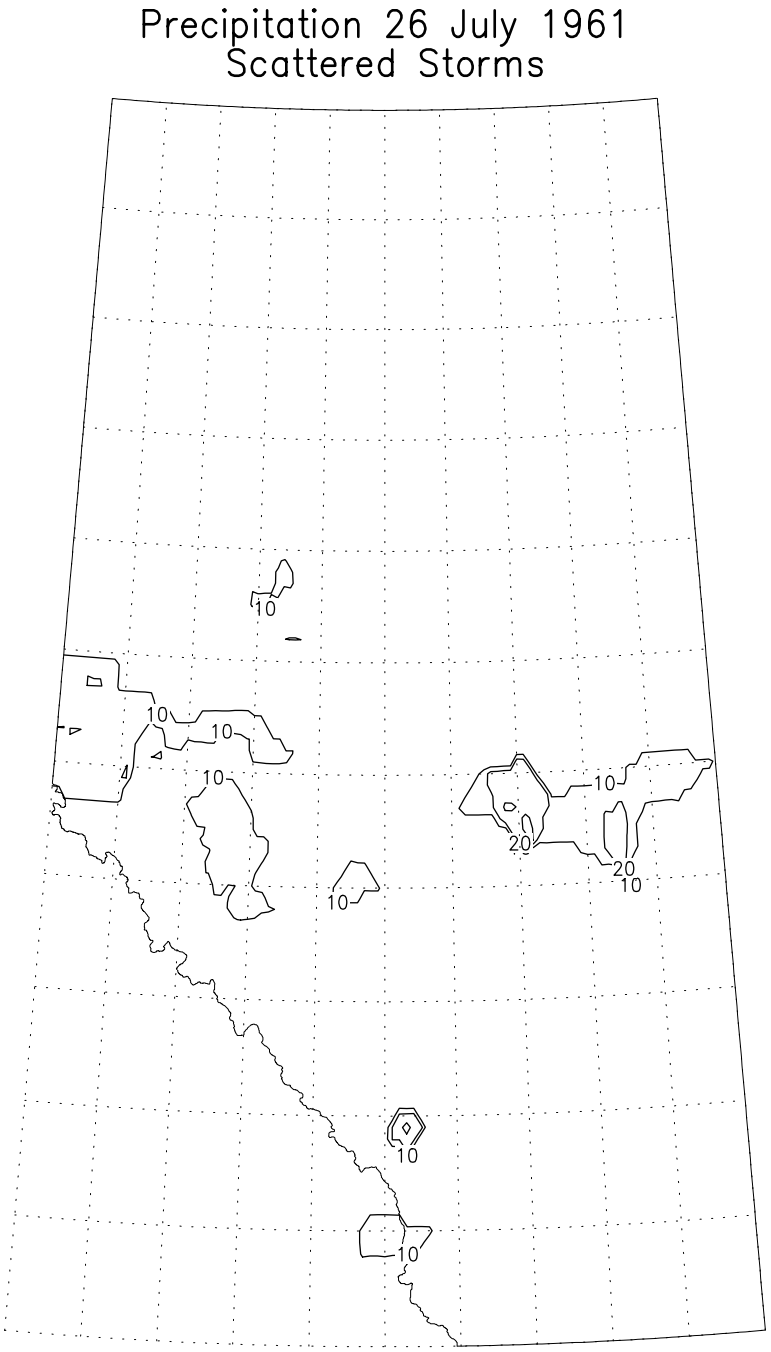

Figure 5.8: Precipitation as interpolated by the hybrid method for small, scattered storms on 26 July 1961.

to directly assess the errors of the polygon data. A rough estimate of the data error of a polygon is given by the above grid-point mean square error divided by $\sqrt{n}$, where n is the spatial degrees of freedom of the climate field over a polygon. This is 1.0 for a small polygon, 2.0 for a large polygon of two independent grid points, and 3.0 for an even larger polygon of three independent grid points. However, this is only a rough estimate and knowing exactly how many independent grid points are within a polygon is not a trivial task. It can be safely claimed that the upper limit of the polygon data error is 1.8–3.2°C for temperature and 1.8–2.4 mm for precipitation, and the lower limit is half of these amounts.

Chapter 6

Agricultural Applications and Climate Change

The interpolated climate data sets produced in this research have a wide range of applications to the development of sustainable agriculture and detection of agroclimate change. In addition to the output produced for soil quality models and drought management strategies, the data can be extended for other purposes. The principal extension is the calculation of several data derivatives pertaining to agricultural production and research. These include the following quantities.

1. degree-days, accumulative degree-days
2. corn-heat-units (CHU), accumulative corn-heat-units
3. dew-point-temperature
4. frost-free periods (calendar dates)
5. start and end of growing season (calendar dates)
6. length of growing season (number of days)
7. evapotranspiration (ET)

8. accumulative precipitation
9. annual potential evapotranspiration (PET), water deficit, soil water budget
10. annual number of days above 30°C
11. annual number of days below 30°C
12. average number of days with precipitation above 0.2 mm, above 25 mm
13. standardized precipitation index $PI = (P_{actual} - P_{avg})/\text{stddev}(P_{actual})$
14. average wind speed and prevailing direction
15. snowfall (derived through a combination of precipitation and temperature)
16. daily, monthly, and annual normals of the above quantities (as appropriate)

Each of these quantities can be derived from the daily climate parameters included in this thesis.

For example, the concept of growing degree-days, also called heat units, assumes that plant growth is related directly to temperature if there are no other limitations. Plant growth starts when the temperature reaches 5°C for most of the major crops in Alberta. Some, like beans and corn, have a minimum threshold of 10°C. The daily mean temperature is calculated as the average of minimum and maximum temperatures. The degree-day is the amount by which the mean temperature exceeds the threshold. For example, if the mean temperature on a given day is 16°C, then the degree-days above 5°C equals 11 and the degree-days above 10°C equals 6. If the mean temperature is below the threshold, the degree-day value is set to zero, for the plant growth is not set back. The degree-day normals allow for the comparison of geographical areas with respect to their growing potential. The

corn-heat-unit is another value which measures growth potential. Heating degree-days relate the daily mean temperature to the threshold of 18°C to indicate the amount of energy needed to heat buildings.

Operational definitions for some of the above derived units can be found in Shen et al. (2000). Such quantities can be calculated as daily data, but also summarized as monthly or annual totals and averages. The monthly and annual forms are suitable for the production of digital maps to comprise an agroclimatic atlas, such as the one for 1951–80 by Dzikowski and Heywood (1990). Furthermore, the entire 1901–2000 data set can be summarized by climate normal periods, 1901–30, 1910–40, $\cdots$, 1971–2000. The 30-year normals of the derived units also can be computed and mapped. This facilitates the visual comparison of the agroclimate between every pair of 30-year normals.

For drought risk management, an important statistic which can be derived from the data set is the precipitation frequency. Essentially, the weekly precipitation amounts are divided into bins, and occurrences of the amounts are listed as a percent, much like a histogram. These percentages are then used to calculate the probability of drought given a particular history of precipitation up to the week in question. This is the idea behind the development of a drought indicator, currently being studied by AAFRD, Conservation and Development Branch.

Climate change is an important issue related to agriculture and ecological sustainability. The concept of climate change is often associated with temperature, precipitation, and CO_2 levels both by the media and in general conversation. But increasing levels of atmospheric CO_2, combined with an increase in nighttime temperatures, affect plant growth (biomass) and water usage (evapotranspiration). These changes should be studied, not only in the base units of temperature and precipitation, but also in the derived units applicable to agriculture.

In summary, the interpolated, 100-year, daily climate data from 1 January 1901 to 31 December 2000 can be used as the master data set for various kinds of agricultural and climatic research and applications. The data set is the first of its kind in Alberta. In applying the data set, users should be cautioned against assuming there are lots of data everywhere in the province. This study will help to identify critical regions where enhanced observations are needed and regions where redundant stations may be removed. Our interpolation method retains not only the accuracy of means but also spatial and temporal variance. The method has avoided the problem of over-smoothing found in most interpolation products (Higgins et al., 1996; Xie and Arkin, 1997). Hence our resulting data are more realistic for soil quality modelling and digital farming. It is our opinion that other interpolated data sets should be examined for the problem of temporal and spatial variances and revised if necessary. The Climate Prediction Center of NOAA is using our method to revise its 1°-by-1° interpolated daily precipitation data.

References

[1] Agriculture Canada, Ecological Stratification Working Group, 1995: A National Ecological Framework for Canada. Agriculture and Agri-Food Canada, Research Branch, Centre for Land and Biological Resources Research and Environment Canada, State of the Environment Directorate, Ecozone Analysis Branch, Ottawa/Hull. Report and National map at 1:7 500 000 scale.

[2] Bootsma, A., and M. Ballard, 1999: A National Ecological Framework for Canada, Canadian EcoDistrict Climate Normals 1961–1990. `http://sis.agr.gc.ca/cansis/nsdb/ecostrat/climate_normals_1961-90.html`.

[3] Changnon, S.A., and K.E. Kunkel, 1999: Rapidly expanding uses of climate data and information in agriculture and water resources: causes and characteristics of new applications, *Bull. Amer. Meteor. Soc.*, **80**, 821–830.

[4] Cressie, N., 1993: *Statistics for Spatial Data*, Wiley, 900 pp.

[5] Dowd, K., and C. Severance, 1998: *High Performance Computing, 2nd ed.*, O'Reilly & Associates, Inc., 446 pp.

[6] Dzikowski, P., and R. Heywood, 1990: *Agroclimatic Atlas of Alberta*, Alberta Agriculture, Food and Rural Development, Agdex 071-1, 31 pp.

[7] Environment Canada, Atmospheric Environment Service, 1982a: *Canadian Climate Normals 1951–1980, Volume 1, Solar Radiation,* 57 pp.

[8] Environment Canada, Atmospheric Environment Service, 1982b: *Canadian Climate Normals 1951–1980, Volume 5, Wind,* 283 pp.

[9] Environment Canada, Atmospheric Environment Service, 1993: *Canadian Climate Normals 1961–1990, Prairie Provinces,* 266 pp.

[10] Flanagan, D.C., and S.J. Livingston, 1995: USDA Water Erosion Prediction Project: Version 95.7 user summary. NSERL report No. 11. USDA-ARS National Erosion Research Laboratory, West Lafayette, Indiana, U.S.A.

[11] Haining, R., 1990: *Spatial Data Analysis in the Social and Environmental Sciences*, Cambridge University Press.

[12] Hansen, J., and S. Lebedeff, 1987: Global trends of measured surface air temperature, *J. Geophys. Res.*, **92**, 13,345–13,372.

[13] Higgins, W., J.E. Janowiak, and Y.-P. Yao, 1996: A gridded hourly precipitation data base for the United States (1963–1993), *NCEP/Climate Prediction Center ATLAS*, No. 1, Camp Springs, MD 20746.

[14] Huff, F.A., and W.L. Shipp, 1969: Spatial correlations of storms, monthly and seasonal precipitation, *J. Appl. Meteo.*, **8**, 542–550.

[15] Isaaks, E.H., and R.M. Srivastava, 1989: *An Introduction to Applied Geostatistics*, Oxford University Press, 561 pp.

[16] Karl, T.R., and R.W. Knight, 1998: Secular trends of precipitation amount, frequency, and intensity in the United States, *Bull. Amer. Meteoro. Soc.*, **79**, 231–241.

[17] Kruse, R.L., 1987: *Data Structures and Program Design, 2nd ed.*, Prentice-Hall, Inc., 586 pp.

[18] Mackey, B.G., D.W. McKinney, Y.-Q. Yang, J.P. McMahon, and M.F. Hutchinson, 1996: Site regions revisited: a climatic analysis on Hill's site regions for the province of Ontario using a parametric method, *Can. J. For. Res.*, **26**, 333–354.

[19] Osborn, T.J., and M. Hulme, 1997: Development of a relationship between station and grid-box rainday frequency for climate model evaluation, *J. Clim.*, **10**, 1885–1908.

[20] Sharpley, A.N., and J.R. Williams, eds. 1990: EPIC - Erosion / Productivity Impact Calculator: 1. Model Documentation. U.S. Department of Agriculture Technical Bulletin No. 1768, 235 pp.

[21] Shen, S.S.P., K. Cannon, and G. Li, 2000: Alberta 1961–1997 climate data on EDP and SLC polygons: data derivatives and data formation for soil quality models. Alberta Agriculture, Food and Rural Development, Research Report.

[22] Shen, S.S.P., P. Dzikowski, G. Li, and D. Griffith, 2001: Interpolation of 1961–1997 Daily Temperature and Precipitation Data onto Alberta Polygons of EcoDistrict and Soil Landscapes of Canada, *J. Appl. Meteo.*, **40**, 2162–2177.

[23] Shields, J.A., C. Tarnocai, K.W.G. Valentine, and K.B. MacDonald. 1991: Soil landscapes of Canada: Procedures manual and user's handbook. Land Resource Research Centre, Research Branch, Agriculture Canada, Ottawa, Ontario (Publication 1868/E). 74 pp.

[24] Tøndering, C., 2001: History and Information on the Christian Calendar (Julian). `http://webexhibits.org/calendars/calendar-christian.html`.

[25] U.S. Naval Observatory. Astronomical Applications Department, 2001: Julian date converter. `http://aa.usno.navy.mil/data/docs/JulianDate.html`.

[26] Xie, P., and P.A. Arkin, 1997: Global Precipitation: A 17-year monthly analysis based on gauge observations, satellite estimates, and numerical model outputs, *Bull. Amer. Meteo. Soc.*, **78**, 2539–2558.

Appendix A

Temporal Interpolations on the Radiation Data

For the radiation interpolation, a total of 29 dates were missing from the entire combined data set, so a temporal interpolation was performed to fill in the gaps for two station files thus making the entire data run complete. The missing blocks of dates were as follows.

For station 3033890:

```
1996  4  6
1996  4  7
1996  4  8

1996  4 11
1996  4 12
```

For station 301222F:

```
1996  4 15        1997  1 25
1996 10 16
1996 10 17        1997  7 16
1996 10 18
1996 10 19        1998  6 30
1996 10 20
1996 10 21        1999  3  2
1996 10 22        1999  3  3
1996 10 23        1999  3  4
1996 10 24        1999  3  5
```

```
1996 10 25
1996 10 26
1996 10 27
1996 10 28
1996 10 29
1996 10 30
1996 10 31
```

Appendix B

Julian Day Code

```
 1 c
 2 c $Id: julian.f,v 1.1 2001/11/07 18:24:24 darren Exp $
 3 c
 4 c  small program to demonstrate and test the code for converting
 5 c  to and from Julian Days.
 6 c
 7 c  Darren Paul Griffith.  November 2001.
 8 c  code taken from the website:
 9 c  http://webexhibits.org/calendars/calendar-christian.html
10
11       implicit none
12
13       integer whichyear
14       integer i
15       integer year, month, day, jday, reference
16
17       integer JD
18
19       reference = 2451910     ! Dec 31, 2000
20       do jday = reference-1 , reference+1
21
22         call fromJD(jday,year,month,day)
23         print 24, jday, year, month, day
24   24    format(I12,3I8)
25
26       enddo
27
28       stop
29       end
30
31
32 ccccccccccccc-------subroutines--------ccccccccccccccccc
33
34       integer function JD (year, month, day)
35
36 c       converts {year,month,day} to a julian day  (for AD dates only)
37 c       taken from webpage:
38 c         http://webexhibits.org/calendars/calendar-christian.html
39 c       divisions are integer divisions
40 c
41 c       results verified for 1 Jan 1901, 31 Dec 2000, 1 Jan 1961, and 31 Dec 1997
```

```
 42
 43       implicit none
 44       integer year,month,day
 45       integer a, y, m
 46
 47       a = (14-month)/12
 48       y = year+4800-a
 49       m = month + 12*a - 3
 50
 51       !  For a date in the Gregorian calendar:
 52       JD = day + (153*m+2)/5 + y*365 + y/4 - y/100 + y/400 - 32045
 53
 54 c      !  For a date in the Julian calendar:
 55 c      JD = day + (153*m+2)/5 + y*365 + y/4 - 32083
 56
 57
 58       return
 59       end
 60
 61
 62 ccccccccccccc-------subroutines--------ccccccccccccccccc
 63
 64       subroutine fromJD (jd, year, month, day)
 65
 66 c       converts a julian day to {year,month,day}
 67 c         (for the gregorian calendar only)
 68 c       taken from webpage:
 69 c         http://webexhibits.org/calendars/calendar-christian.html
 70 c       divisions are integer divisions
 71 c
 72 c       results verified for 1 Jan 1901, 31 Dec 2000, 1 Jan 1961, and 31 Dec 1997
 73 c          and the year 2000 does give a leap year, and 2001 does not
 74
 75       implicit none
 76       integer jd, year, month, day
 77       integer a, b, c, d, e, m
 78
 79       ! gregorian calendar
 80       a = JD + 32044
 81       b = (4*a+3)/146097
 82       c = a - (b*146097)/4
 83
 84 c      ! julian calendar
 85 c      b = 0
 86 c      c = JD + 32082
 87
 88       ! for both calendars
 89       d = (4*c+3)/1461
 90       e = c - (1461*d)/4
 91       m = (5*e+2)/153
 92
 93       day = e - (153*m+2)/5 + 1
 94       month = m + 3 - 12*(m/10)
 95       year = b*100 + d - 4800 + m/10
 96
 97       return
 98       end
 99
100
```

Appendix C

Wind Interpolation Code

This appendix contains an abridged version of the wind interpolation code, which implements the nearest-station assignment algorithm.

```
c
c $Id: wind150.f,v 1.16 2002/01/04 23:16:13 darren Exp $
c


ccccccccccc--------declarations--------------ccccccccccccccccccccccc

      implicit none

      integer DAYOFFSET, FIRSTDAY, LASTDAY,NDAYS
      parameter(FIRSTDAY  = 2432552)         ! 01 Jan 1948
      parameter(NDAYS     =   19359)         ! 100 years
      parameter(DAYOFFSET = FIRSTDAY-1)      ! the day before 01 Jan 1901
      parameter(LASTDAY   = FIRSTDAY+NDAYS)  ! 31 Dec 2001

      integer FIRST,LAST
      parameter(FIRST = 1)        ! where 01 Jan 1961 occurs in 1..NDAYS
      parameter(LAST  = NDAYS)

      Integer NSTATIONS,NPOLY,NPOINTS,LASTPOLYID
      parameter(NSTATIONS=518,NPOLY=149,NPOINTS=6633)    ! edp 1901-2000
      parameter(LASTPOLYID=1019)
c      parameter(NSTATIONS=518,NPOLY=894,NPOINTS=6746)    ! slc 1901-2000
c      parameter(LASTPOLYID=5130)

c     *** MAKE SURE to change the grid and polygon filenames from edp to slc, etc.
c         and to change the station files from 6190 to 9197.

      double precision pi
      parameter(pi = 3.14159265358979323846264338327 95)

      character*1  INTERPTYPE
      parameter(INTERPTYPE='w')

      integer  tempb, tempc
      character*16 tempname
      character*15 polyID_char(NPOLY)         ! gives the polygon id (filename) given the index
      integer      polyID_int(NPOLY)          ! gives the polygon id (integer)  given the index
      integer      polyID_idx(LASTPOLYID)     ! gives the index given the polygon id
      character*15  stationID(NSTATIONS)

      integer i, j, k

      double precision station_lat(NSTATIONS), station_lon(NSTATIONS)

      double precision point_in_poly(NPOINTS)   ! returns polygon id which contains that grid point
      double precision point_lat(NPOINTS), point_lon(NPOINTS)  ! grid point latitude and longitude
      character*10 pointstatus(NPOINTS)            ! this is read from the file but is not used
      integer tempid
```

```
51
52       double precision dtable(NSTATIONS,NPOINTS)  ! distance to each station for every grid point
53       integer sorted_stations(NSTATIONS,NPOINTS)  ! for each grid point, this gives a list of the
54                                                   ! stations (indices) from closest to farthest
55
56       integer m, max_idx, tempint
57       integer tempspeed_int, tempdir_int
58       double precision tempreal, tempspeed, tempdir, u, v
59
60       double precision stationdata_uv(2,NSTATIONS)    ! 2 := wind vector components u and v
61       double precision griddata_uv(2,NPOINTS)         ! 2 := wind vector components u and v
62
63       integer day, nearest, index
64
65       double precision polydata_uv(2,NDAYS,NPOLY)     ! holds the averaged data for all days
66       integer num_points_in_poly(NPOLY)               ! number of grid points involved in the
67                                                       ! averaging for that polygon
68
69       double precision a
70       integer tempday, month, year
71
72       integer record_day(NDAYS)        ! index file stuff
73       integer record_start(NDAYS)
74       integer record_end(NDAYS)
75       integer recdays, recstart, recend
76
77
78
79       ! subroutine declarations
80       double precision distance
81       double precision convert
82
83
84 ccccccccccc--------binary file openning and indexing--ccccccccccccccccc
85
86       ! binary station data file
87       open(11,file='/agr/nov2001CD/binaries/wind.bin',
88      &    access='direct', form='unformatted',recl=3*8)
89
90       ! index file for binary file {day, record number of beginning, end}
91       open(12,file='/agr/nov2001CD/binaries/wind.idx',
92      &    form='formatted')
93
94       ! skip first line (by reading it) because it's junk
95       read(12,25) record_day(1), record_start(1), record_end(1)
96       ! thus record_day(    1) is going to be equal to FIRSTDAY
97       ! and  record_day(NDAYS) is going to be equal to  LASTDAY
98
99       do i=1,NDAYS
100        read(12,25) record_day(i), record_start(i), record_end(i)
101      enddo
102      close(12)
103   25 format(3I12)
104
105
106 ccccccccccc--------read the station infomation---------ccccccccccccccccc
107
108      ! station latitude and longitude in integer format
109      open(10,file='windstationinfo_1901.prn',form='formatted')
110      do i=1,NSTATIONS
111        read(10,21) stationID(i), tempb, tempc      ! integer format of lat/lon
112        station_lat(i) = convert(tempb)             !   convert it to decimal lat/lon
113 station_lon(i) = convert(tempc)
114      enddo
115      close(10)
116  21  format(A7,I5,I6)
117
118      ! grid point locations for each polygon
119      open(15,file='gridedp_sorted.dat')
120      do i=1,NPOINTS
121        read(15,26) tempid, pointstatus(i), point_lat(i), point_lon(i)
122        point_in_poly(i)=tempid
123      enddo
124      close(15)
125
126  26  format(I4,A10,F13.2,F15.2)
127
128      do i=1,LASTPOLYID
129        polyID_idx(i) = -999.9            ! invalid polygon id's get a -999.9
130      enddo
131
```

```
      open(14,file='polyID_info_edp.prn')                  ! output filenames
      do i=1,NPOLY
        read(14,24) polyID_char(i), polyID_int(i)
polyID_idx(polyID_int(i)) = i         ! store the index which points to polygon id
      enddo
      close(14)

 24   format(A7,I7)


ccccccccccc------calculate the distance table-----------ccccccccccc

      write(0,*) 'Calculating distance table...'
      do j=1,NPOINTS
      do i=1,NSTATIONS
        dtable(i,j) = distance(station_lat(i), station_lon(i),
     &                         point_lat(j), point_lon(j))
        sorted_stations(i,j) = i   ! store the i index, which gets swapped later
      enddo
      enddo


ccccccccccc-----sort the distance table------ccccccccccccccccccccccccccccccc

      write(0,*) 'Sorting the distance table...'

c     We are sorting the distance table dtable(i,j) with j fixed.
c     Use selection sort, which looks for the largest distance and places
c     it in position.

      do j=1,NPOINTS          ! fix your grid point

c----selection sort begins here-----------------------------------------------
      do i=NSTATIONS,2,-1
        ! find the max distance from 1 to i
        m = 1
        do k=2,i
          if (dtable(m,j).lt.dtable(k,j)) then
            m = k
          endif
        enddo
        max_idx=m

        ! swap max_idx with the guy who is there

        tempreal = dtable(max_idx,j)
        dtable(max_idx,j) = dtable(i,j)
        dtable(i,j)= tempreal

        tempint = sorted_stations(max_idx,j)
sorted_stations(max_idx,j) = sorted_stations(i,j)
sorted_stations(i,j) = tempint

! sorted_stations(1,j) now contains the index of the station nearest to grid point j

      enddo
c----selection sort ends here-------------------------------------------------

      ! print the distance table, just to be sure


      enddo      ! end all grid points


ccccccccccccc-----------nearest-station assignment-----------------ccccccccccccccccc

  ! for every day...
    ! assign to each grid point...
      ! the value of the nearest station that has data (that is, not -999.9, -999.9)

      write(0,*) 'Begin nearest-station assignment and averaging...'

      ! compiler complained when I tried to store points and days in one array
      ! so we have to do one day at a time and not store the intermediate response

      ! initialization loop
      do i = 1, NPOLY
      do day = 1, NDAYS
polydata_uv(1,day,i) = 0.0           ! this array MUST be empty to use it
polydata_uv(2,day,i) = 0.0
      enddo
```

```
213       enddo
214       write(0,*) '  Done initialization loop.'
215
216       do day = FIRST,LAST
217
218         ! initialize stationdata array; stations without data will be -999.9
219 do i=1,NSTATIONS
220   stationdata_uv(1,i)=-999.9    ! u
221   stationdata_uv(2,i)=-999.9    ! v
222 enddo
223
224 recstart=record_start(day)
225 recend=record_end(day)
226 recdays=recend-recstart+1
227
228 do i=1,recdays
229   read(11,rec=recstart+i-1) a,tempspeed,tempdir
230           index=int(a)              ! get the station index
231   if ((tempspeed.gt.-99.9).or.(tempdir.gt.-99.9)) then   ! check for missing values
232             tempdir   = (90 - tempdir)* pi / 180.0
233             stationdata_uv(1,index) = tempspeed * cos(tempdir)
234             stationdata_uv(2,index) = tempspeed * sin(tempdir)
235   else
236             stationdata_uv(1,index) = tempspeed          ! -999.9
237             stationdata_uv(2,index) = tempdir            ! -999.9
238   endif
239 enddo
240
241
242
243       do i = 1,NPOINTS
244
245         !***** look until you find a station with data that day *****
246         nearest = 0
247  200    nearest = nearest + 1               ! 1 is the closest, 2 is the 2nd closest,...
248         index = sorted_stations(nearest,i)   ! get the index for the station
249         if (stationdata_uv(1,index).lt.-99.9) then
250           goto 200
251         endif
252
253         if (nearest.gt.NSTATIONS) then
254           write(0,*) '***ERROR: number of stations exceeded.
255      &             No data on that day?***'
256           goto 999   ! exit the entire program
257         endif
258
259         ! assign the data for that day
260         griddata_uv(1,i) = stationdata_uv(1,index)
261         griddata_uv(2,i) = stationdata_uv(2,index)
262
263 c        print 32, i, nearest, dtable(nearest, i)   ! useful for printing which distances
264 c  32    format(2I8,F8.2)                           ! occur in grid point nearest-station
265                                                     ! assignment
266
267         !***** end of search ************************************
268
269       enddo  ! end over all points
270
271
272       ! ***** average points over all polygons
273
274       ! intialization loop
275       do i=1,NPOLY
276         num_points_in_poly(i) = 0
277       enddo
278
279       do i=1,NPOINTS
280         index = polyID_idx(point_in_poly(i))       ! gets the polygon index from the polygon id
281         polydata_uv(1,day,index) = polydata_uv(1,day,index) +
282      &                             griddata_uv(1,i)         ! running total for the average
283         polydata_uv(2,day,index) = polydata_uv(2,day,index) +
284      &                             griddata_uv(2,i)
285 num_points_in_poly(index) = num_points_in_poly(index) + 1  ! keep track of how many
286       enddo                                                   ! we're adding
287
288       if(mod(day,1000).eq.0) then
289         write(0,35) day
290       endif
291   35  format('   day: ', I8)
292
293       ! finish the averaging by dividing through
```

```
294       do i=1,NPOLY
295         polydata_uv(1,day,i) = polydata_uv(1,day,i) /
296      &                         real(num_points_in_poly(i))
297         polydata_uv(2,day,i) = polydata_uv(2,day,i) /
298      &                         real(num_points_in_poly(i))
299       enddo
300
301
302       enddo  ! end over all days
303
304
305 ccccccccccccc--------produce the output ------------------ccccccccccccccccc
306
307       write(0,*) 'Processing and writing final output...'
308
309       do i=1,NPOLY
310         tempname=INTERPTYPE // polyID_char(i)    ! puts a single character before the filename
311         open(18,file=tempname)                   ! open the output file for each polygon
312 write(0,34) i, polyID_int(i)
313   34    format('   Polygon: ',2I8)
314
315       do day=FIRST,LAST
316
317         call fromJD(day+DAYOFFSET,month,tempday,year)
318
319  u = polydata_uv(1,day,i)
320  v = polydata_uv(2,day,i)
321          tempspeed = sqrt(u**2+v**2)
322  tempdir = 90.0 - (datan2(v,u) * 180.0 / pi)
323  if (tempdir.lt.0.0) then
324    tempdir = tempdir + 360.0
325  endif
326  tempspeed_int = nint(tempspeed*10.0)
327          tempdir_int = nint(tempdir)      ! round direction to nearest whole number
328
329          if (tempspeed_int.eq.0) then      ! speed of 0 should have direction of 0 as well
330    tempdir_int = 0
331  endif
332
333  ! final output to each polygon file
334          write(18,33) year, month, tempday, tempspeed, tempdir_int
335   33     format(I4,2I3,F6.1,I4)
336
337
338
339        enddo    ! end over all days
340
341
342        close(18)     ! close the output for each polygon
343       enddo   ! end over all polygons
344
345
346
347
348 ccccccccccccc--------------------------ccccccccccccccccc
349
350       close(11)  ! binary station data file
351
352 999   stop
353       end
354
355
356
357 ccccccccccccc-------subroutines--------ccccccccccccccccc
358
359       double precision function convert(a)
360 c          converts the integer format latitude/longitude from dddmm to
361 c          decimal degrees ddd.dddd
362         implicit none
363         integer a, degrees, minutes
364
365 degrees = int(real(a)/100.0)
366 minutes = a - int(real(a)/100.0)*100
367
368 convert = real(degrees) + real(minutes)/60.0
369
370         return
371       end
372
373
374 ccccccccccccc-------subroutines--------ccccccccccccccccc
```

```
375
376       double precision function distance(ay, ax, by, bx)
377 c          gives great circle distance in km from two points (lat N, long W)
378 c          in decimal degrees, based on a spherical model of the earth
379         implicit none
380         double precision ax, ay, bx, by
381         double precision aax, aay, bbx, bby
382         double precision pi, k
383
384         pi = 3.14159265358979323846264338327 95
385         k = 111.120                                    ! km/(degree of latitude)
386
387         aax = ax * pi/180.0                            ! convert to radians
388         aay = ay * pi/180.0
389         bbx = bx * pi/180.0
390         bby = by * pi/180.0
391
392         distance = k * 180.0 / pi * acos(sin(aay)*sin(bby)
393      &             + cos(aay)*cos(bby)*cos(aax-bbx))
394
395         return
396       end
397
```

Appendix D

Temperature Interpolation Code

This appendix contains an abridged version of the temperature interpolation code, which implements the inverse-distance algorithm.

```
 1 c
 2 c $Id: interp_dpg.f,v 1.12 2002/01/02 06:19:32 darren Exp $
 3 c
 4 c  duck///home/darren/climate/temperature/interp_dpg.f
 5 c
 6
 7 ccccccccccc--------declarations--------------ccccccccccccccccccccccc
 8
 9       implicit none
10
11       integer DAYOFFSET, FIRSTDAY, LASTDAY,NDAYS
12       parameter(FIRSTDAY  = 2415386)         ! 01 Jan 1901
13       parameter(NDAYS     =   36525)         ! 100 years
14       parameter(DAYOFFSET = FIRSTDAY-1)      ! the day before 01 Jan 1901
15       parameter(LASTDAY   = FIRSTDAY+NDAYS)  ! 31 Dec 2000
16
17       integer FIRST,LAST
18       parameter(FIRST = 1)        ! where the starting date you want occurs in 1..NDAYS
19       parameter(LAST  = NDAYS)
20
21       integer NSTATIONS,NPOLY,NPOINTS,LASTPOLYID
22 c      parameter(NSTATIONS=2611,NPOLY=149,NPOINTS=6633)    ! edp 1901-2000
23 c      parameter(LASTPOLYID=1019)
24       parameter(NSTATIONS=2611,NPOLY=894,NPOINTS=6746)    ! slc 1901-2000
25       parameter(LASTPOLYID=5130)
26
27 c     *** MAKE SURE to change the grid and polygon filenames from edp to slc, etc.
28 c         and to change the station info file from 6190 to 9197.
29
30       character*18 GRIDSTATS_FILENAME
31 c      parameter(GRIDSTATS_FILENAME='gridstats_tedp.dat')
32       parameter(GRIDSTATS_FILENAME='gridstats_tslc.dat')
33
34
35       double precision LENGTHSCALE
36       parameter(LENGTHSCALE = 200.0)   ! 200.0 km length scale for the temperature field
37
38       character*1  INTERPTYPE
39       parameter(INTERPTYPE='t')
40
41       integer i, j, k
42
43       integer tempb, tempc
44       character*16 tempname
45       character*15 polyID_char(NPOLY)         ! gives the polygon id (filename) given the index
46       integer      polyID_int(NPOLY)          ! gives the polygon id (integer)  given the index
```

```
47       integer      polyID_idx(LASTPOLYID)     ! gives the index given the polygon id
48       character*15  stationID(NSTATIONS)
49
50       double precision station_lat(NSTATIONS), station_lon(NSTATIONS)
51
52       double precision point_in_poly(NPOINTS)  ! returns polygon id which contains that grid point
53       double precision point_lat(NPOINTS), point_lon(NPOINTS)  ! grid point latitude and longitude
54       character*10 pointstatus(NPOINTS)                   ! read from file but not used
55       integer tempid
56
57
58       double precision dtable(NSTATIONS,NPOINTS)  ! distance to each station for every grid point
59       integer sorted_stations(NSTATIONS,NPOINTS)  ! for each grid point, this gives a list of
60                                                   ! stations (indices) from closest to farthest
61
62       integer m, max_idx, tempint
63       double precision tempreal
64
65       double precision stationdata(2,NSTATIONS) ! for a single day: tmax, tmin
66       double precision griddata_nearest(2,NPOINTS)     ! can't hold NDAYS*NPOINTS -- too big !
67       double precision griddata_invdist(2,NPOINTS)     ! can't hold NDAYS*NPOINTS -- too big !
68       double precision invdiststation(2,8)    ! holds the data for the 8 nearest stations with data
69       double precision invdistdistance(8)     ! holds the distance for the 8 nearest stations
70
71       integer year, month, tempday
72       integer day, station, datatype, nearest, index        ! loop indicies
73       integer counter, stationcounter
74
75       double precision polydata(2,NDAYS,NPOLY)   ! holds the averaged data for all days
76       integer num_points_in_poly(NPOLY)          ! number of grid points involved in
77                                                  ! the averaging for that polygon
78
79       double precision a,b,c,d        ! numerator and denominator in inverse-distance formula
80       integer gridstats(9)   ! stores a histogram count for the number of stations used for EACH
81                              ! grid point (can be 0 to 8, which gets translated to 1 to 9 in
82                              ! this array)
83
84       integer record_day(NDAYS)        ! index file stuff
85       integer record_start(NDAYS)
86       integer record_end(NDAYS)
87       integer recdays, recstart, recend
88
89       ! subroutine declarations
90       double precision distance
91       double precision convert
92
93
94 ccccccccccc--------binary file openning and indexing--ccccccccccccccccc
95
96       ! binary station data file
97       open(11,file='/agr/nov2001CD/binaries/temppcpn.bin',
98      &    access='direct', form='unformatted',recl=4*8)
99
100      ! index file for binary file {day, record number of beginning, end}
101      open(12,file='/agr/nov2001CD/binaries/temppcpn.idx',
102     &    form='formatted')
103
104      ! skip first line (by reading it) because it's junk
105      read(12,25) record_day(1), record_start(1), record_end(1)
106      ! thus record_day(    1) is going to be equal to FIRSTDAY
107      ! and  record_day(NDAYS) is going to be equal to  LASTDAY
108
109      do i=1,NDAYS
110        read(12,25) record_day(i), record_start(i), record_end(i)
111      enddo
112      close(12)
113   25 format(3I12)
114
115
116 ccccccccccc--------read the station infomation---------ccccccccccccccccc
117
118      ! station latitude and longitude in integer format
119      open(10,file='stationinfo_1901.prn',form='formatted')
120      do i=1,NSTATIONS
121        read(10,21) stationID(i), tempb, tempc      ! integer format of lat/lon
122        station_lat(i) = convert(tempb)             !   convert it to decimal lat/lon
123        station_lon(i) = convert(tempc)
124      enddo
125      close(10)
126   21 format(A7,I5,I6)
127
```

```
128       ! grid point locations for each polygon
129       open(15,file='gridslc_sorted.dat')
130       do i=1,NPOINTS
131         read(15,26) tempid, pointstatus(i), point_lat(i), point_lon(i)
132         point_in_poly(i)=tempid
133       enddo
134       close(15)
135   26   format(I4,A10,F13.2,F15.2)
136
137
138       do i=1,LASTPOLYID
139         polyID_idx(i) = -999.9       ! invalid polygon id's get a -999.9, just to be safe
140       enddo
141
142
143       open(14,file='polyID_info_slc.prn')       ! output filenames
144       do i=1,NPOLY
145         read(14,24) polyID_char(i), polyID_int(i)
146         polyID_idx(polyID_int(i)) = i           ! store the index which points to polygon id
147       enddo
148       close(14)
149
150   24   format(A7,I7)
151
152
153 ccccccccccc------calculate the distance table-----------ccccccccccc
154
155       write(0,*) 'Calculating distance table...'    ! write messages to standard error
156       do j=1,NPOINTS
157       do i=1,NSTATIONS
158         dtable(i,j) = distance(station_lat(i), station_lon(i),
159      &                         point_lat(j), point_lon(j))
160
161         if (dtable(i,j).le.0.001) then     ! we want to eliminate
162           dtable(i,j) = 0.001              ! division-by-zero errors for
163         endif                                    ! inverse-distance formula
164                                            ! when grid points are on top
165                                            ! of stations (d=0)
166
167
168         sorted_stations(i,j) = i   ! store the i index, which gets swapped later
169
170       enddo
171       enddo
172
173
174 ccccccccccc-----sort the distance table------ccccccccccccccccccccccccccccccc
175
176       write(0,*) 'Sorting the distance table...'
177
178 c     We are sorting the distance table dtable(i,j) with j fixed.
179 c     Use selection sort, which looks for the largest distance and places
180 c     it in position.
181
182       do j=1,NPOINTS          ! fix your grid point
183
184 c----selection sort begins here-----------------------------------------------
185       do i=NSTATIONS,2,-1
186         ! find the max distance from 1 to i
187         m = 1
188         do k=2,i
189           if (dtable(m,j).lt.dtable(k,j)) then
190             m = k
191           endif
192         enddo
193         max_idx=m
194
195         ! swap max_idx with the guy who is there
196
197         tempreal = dtable(max_idx,j)
198         dtable(max_idx,j) = dtable(i,j)
199         dtable(i,j)= tempreal
200
201         tempint = sorted_stations(max_idx,j)
202         sorted_stations(max_idx,j) = sorted_stations(i,j)
203         sorted_stations(i,j) = tempint
204
205         ! sorted_stations(1,j) now contains the index of
206         !   the station nearest to grid point j
207
208       enddo
```

```
c----selection sort ends here-------------------------------------------------

      if(mod(j,1000).eq.0) write(0, 36) j
 36   format('   completed grid point: ', I8)


      enddo      ! end all grid points


ccccccccccccc-------initialization loop------------ccccccccccccccccc

      write(0,*) 'Begin initialization loop of polygon data space...'
      do i = 1, NPOLY
      do day = 1, NDAYS
        polydata(1,day,i) = 0.0      ! this array MUST be empty to use it
        polydata(2,day,i) = 0.0
      enddo
      enddo

      do i=1,9
        gridstats(i) = 0
      enddo


ccccccccccccc-----------inverse distance interpolation-----------------ccccccccccccccccc

   ! for every day...
     ! assign to each grid point...
      ! the distance-weighted average of the values of the
      !   8 (or less) nearest stations
      !   that have data (that is, not -999.9, -999.9)

 777  write(0,*) 'Begin inverse distance-interpolation and averaging...'

      ! compiler complained when I tried to store points and days in one array
      ! so we have to do one day at a time and not store the intermediate response

      do day = FIRST,LAST

        ! initialize stationdata array; stations without data will be -999.9
        do i=1,NSTATIONS
          stationdata(1,i)=-999.9   ! tmax
          stationdata(2,i)=-999.9   ! tmin
        enddo

        recstart=record_start(day)
        recend=record_end(day)
        recdays=recend-recstart+1

        do i=1,recdays
          read(11,rec=recstart+i-1) a,b,c,d
          index=int(a)              ! get the station index
          stationdata(1,index) = b  ! tmax
          stationdata(2,index) = c  ! tmin
        enddo



      do i = 1,NPOINTS

        !***** search until you find 8 stations within 200 km with data that day *****
        counter = 0
        stationcounter = 0
 201    counter = counter + 1               ! 1 is the closest, 2 is the 2nd closest,...
          if (counter.gt.NSTATIONS) goto 202 ! we've exhausted all possible stations so exit
          if (stationcounter.ge.8) goto 202  ! we've found the 8 that we needed so exit

          index = sorted_stations(counter,i)   ! get the index for the station
          if ((stationdata(1,index).lt.-99.9).or.    ! if either tmax
     &       (stationdata(2,index).lt.-99.9)) goto 201   ! or tmin is missing look again

          if (dtable(counter,i).le.LENGTHSCALE) then
            stationcounter = stationcounter + 1        ! we found a usable station
            invdiststation(1,stationcounter) = stationdata(1,index)   ! store its data
            invdiststation(2,stationcounter) = stationdata(2,index)
             invdistdistance(stationcounter) =  dtable(counter,i)
          endif

        goto 201

        !***** end of search-until loop  ******************************
```

```
290
291         ! store the number of stations used for each grid point to gridstats.dat (0 to 8)
292  202      gridstats(stationcounter+1) = gridstats(stationcounter+1) + 1
293
294         ! if, after all that searching, no stations were found
295         ! (ie, stationcounter = 0) then we have to resort to
296         ! nearest-station assignment
297           if (stationcounter.eq.0) then
298           !***** look until you find a nearest-station with data that day *****
299           nearest = 0
300   203     nearest = nearest + 1               ! 1 is the closest, 2 is the 2nd closest,...
301             if (nearest.gt.NSTATIONS) then
302               write(0,*) '***ERROR: number of stations exceeded.
303       &               No data on that day?***'
304               write(0,*) 'jday = ', day + DAYOFFSET
305               goto 999   ! exit the entire program
306             endif
307             index = sorted_stations(nearest,i)   ! get the index for the station
308             if ((stationdata(1,index).lt.-99.9).or.    ! if either tmax
309       &        (stationdata(2,index).lt.-99.9)) goto 203   ! or tmin is missing look again
310
311           !***** end of nearest-station search ***************************************
312
313             ! assign the data for that day
314             griddata_invdist(1,i) = stationdata(1,index)
315             griddata_invdist(2,i) = stationdata(2,index)
316
317
318         else  ! stationcounter is not zero
319           ! calculate the inverse-distance estimate for that point for
320           ! that day
321
322           a = 0.0                                          ! numerator for tmax
323           b = 0.0                                          ! numerator for tmin
324           c = 0.0                                          ! denominator for both
325           do station=1,stationcounter
326             a = a + invdiststation(1,station)/invdistdistance(station)
327             b = b + invdiststation(2,station)/invdistdistance(station)
328             c = c + 1/invdistdistance(station)
329           enddo
330           griddata_invdist(1,i) = a/c
331           griddata_invdist(2,i) = b/c
332
333         endif  ! end (if stationcounter = 0)
334
335
336       enddo  ! end over all points
337
338
339       ! ***** average points over all polygons
340
341       ! intialization loop
342       do i=1,NPOLY
343         num_points_in_poly(i) = 0
344       enddo
345
346       do i=1,NPOINTS
347         index = polyID_idx(point_in_poly(i))    ! gets the polygon index from the polygon id
348         polydata(1,day,index) = polydata(1,day,index) +
349       &                         griddata_invdist(1,i)    ! running total for the average
350         polydata(2,day,index) = polydata(2,day,index) +
351       &                           griddata_invdist(2,i)
352         num_points_in_poly(index) = num_points_in_poly(index) + 1    ! keep track of how
353       enddo                                                          ! many we're adding
354
355       if(mod(day,1000).eq.0) write(0, 35) day
356   35  format('   completed day: ', I8)
357
358
359         ! finish the averaging by dividing through
360       do i=1,NPOLY
361         polydata(1,day,i) = polydata(1,day,i) /
362       &                     real(num_points_in_poly(i))
363         polydata(2,day,i) = polydata(2,day,i) /
364       &                     real(num_points_in_poly(i))
365       enddo
366
367
368       enddo  ! end over all days
369
370
```

```
371
372 ccccccccccccc--------produce the output ------------------ccccccccccccccccc
373
374  888  write(0,*) 'Processing and writing final output...'
375
376       do i=1,NPOLY
377         tempname=INTERPTYPE // polyID_char(i)  ! puts a single character before the filename
378         open(18,file=tempname)                 ! open the output file for each polygon
379         write(0,34) i, polyID_int(i)
380   34    format('   Polygon: ',2I8)
381
382       do day = FIRST,LAST
383       call fromJD(day+DAYOFFSET,month,tempday,year)
384
385          write(18,33) year, month, tempday,                    ! final output to
386      &                polydata(1,day,i), polydata(2,day,i)     !   each polygon file
387   33     format(I4,2I3,2F6.1)
388
389
390
391         enddo    ! end over all days
392
393
394         close(18)     ! close the output for each polygon
395       enddo   ! end over all polygons
396
397
398 ccccccccccccc---gridstats----------------------ccccccccccccccccc
399 c  statistical file for reporting histogram information on the
400 c    number of stations used for EACH gridpoint interpolation
401
402       open(19,file=GRIDSTATS_FILENAME)
403       do i=1,9
404         write(19,37) i-1, gridstats(i), real(gridstats(i))/
405      &                      real(NDAYS*NPOINTS)*100.0
406       enddo
407   37  format(2I12,F12.5)
408
409       close(19)   ! close stats file
410
411
412
413 ccccccccccccc--------------------------ccccccccccccccccc
414
415       close(11)  ! binary station data file
416
417 999   stop
418       end
419
420
421
```

Appendix E

Shell Script for Manipulating Station Files

```
#!/bin/sh
#
# Id: check.sh,v 1.4 2001/12/13 22:34:18 darren Exp darren
#
# Darren Paul Griffith.  November 2001.
#
# This shell script manipulates the station files
# from the nov2001CD data to check one file at a time
# for problems such as (tmax < tmin), (tmax > 50), or
# (tmin < -70), etc.


####### list of station files by climate parameter #######
TEMPPCPN='
1010066 1010235 1010595 1010774 1010780 1010960 1010961 ...
2100120 2100160 2100163 2100167 2100174 2100182 2100200 ...
3010080 3010160 3010164 3010175 3010232 3010234 3010301 ...
4007 4013740 4015400 4015867 4018160 4018161 4020 402002 ...
5000 5009 5011 5015 5020 5038 5043 5091 5153 5164 5169 ...
6008 6218 6230 6233 6235 6236 6237 6238 6302 6307 6364 ...
7047 7148 7156 7159 7204 7228 7234 7248 7250 7258 7265 ...
8021 8043 8087 8093 8101 8152 8156 8161 8202 8207 8211 ...
9033 9047 912 9185 9186 9187 9493 9498 9900 '

CANTEMPPCPN='
1010066 1010235 1010595 1010774 1010780 1010960 1010961 ...
2100120 2100160 2100163 2100167 2100174 2100182 2100200 ...
3010080 3010160 3010164 3010175 3010232 3010234 3010301 ...
4013740 4015400 4015867 4018160 4018161 4020020 4020121 ... '

USTEMPPCPN='1017 1080 1081 1202 1217 1231 1272 1336 1342 ... '

WIND='
1012475 1012710 1013998 1014820 1016640 1017101 1017254 ... '

RH='
1012710 1014820 1017101 1017254 1018598 1018610 1018611 ... '
```

```
RAD='
1025C70 1096450 1108487 1125223 1127800 1192940 1192950
2101300 24131 24143
3012208 301222F 3033890 3036240 3070560
4028060 404037Q 4055736 '


######## user defined parameters ###############
# this allows one to specify which climate
# elements to check

DIR='/agr/nov2001CD/temppcpn'

PROGRAM=./check
LIST=$TEMPPCPN

######## end of user defined parameters ######

for variable in $LIST
do
  echo -n  XXXXXX      $variable.txt      XXXXXX
  $PROGRAM < $DIR/$variable.txt
done
```