`libproj4`: A Comprehensive Library of Cartographic Projection Functions (Preliminary Draft)

Gerald I. Evenden

March, 2005

Contents

List of Figures

Chapter 1

Using the libproj4 Library.

Although this cartographic projection library contains a large number of projections the programmatic usage is quite simple. The main burden of usage is the selection and correct usage of the parameters of the individual projections which is, in most cases, a burden placed upon the user, not the programmer. Usage is very similar to I/O programming where a file is *opened* and a structure is returned that is used by various I/O operation routines—a structure that contains all the details related to a particular file. Other similarities with file handling is that more than one projection can be processed concurrently and the structure is *closed* when finished.

1.1 Basic Usage

A cartographic projection is also a mathematical process like functions included in a compiler's mathematics library such as `sin(x)` to compute $\sin x$ and `asin(x)` to compute the inverse, $\arcsin x$ (also referred to as $\sin^{-1} x$). But unlike most mathematical library functions, the forward, P, and inverse, P^{-1}, cartographic projection functions have a multivariate argument and a bivariate return value:

$$(x, y) \leftarrow P(\lambda, \phi, \cdots) \tag{1.1}$$

$$(\lambda, \phi) \leftarrow P^{-1}(x, y, \cdots) \tag{1.2}$$

where x and y are the planar, Cartesian coordinates, usually in meters, and λ and ϕ are the respective longitude and latitude geographic coordinates in radians.

The biggest complication is the type and number of the additional functional arguments constituting the complete argument list. There is always either the Earth's radius or several techniques for defining the Earth's ellipsoid shape as well as specifications for false origins and units of Cartesian measure. Individual projections may have additional parameters that need to be specified. In all cases, it is necessary for the user to refer to the individual projection description for details about the individual projection parameters required.

Because of the large number of selectable projections, each with their own special list of arguments, the following method was chosen to simplify the number of library entries needed by the programmer to the following prototypes defined in the header file `projects.h`:

```
#include <lib_proj.h>

void *pj_init(int nargs, char *args[]);
XY pj_fwd(LP lp, void *PJ);
LP pj_inv(XY xy, void *PJ);
void pj_free(void *PJ);
```

The complexity of this system is not in programmatic usage as described in the following text, but in understanding and properly using the cartographic control parameters.

The procedure `pj_init` must be called first to select and initialize a projection. Parameters for the projection are passed in a manner identical with the normal C program entry point `main`: a count of the number of parameters and an array of pointers to the characters strings containing the parameters. In this case, the parameter strings are those cartographic parameters discussed in the sections describing the individual projections. By using character strings as arguments the selection of the projection and its arguments can be left to the user and thus avoid a great deal of programming time decoding and implementing a traditional argument list.

Upon successful initialization `pj_init` returns a `void` pointer to a data structure that is used as the second argument with the forward, `pj_fwd`, and inverse, `pj_inv`, projection functions. Because the data structure returned by `pj_init` contains all the information for the computing the projection selected by the initialization call, any number of additional initialization calls can be made and used concurrently.

If the initialization call failed then a null value is returned. See Section 1.3 for details on determining cause of failure.

The first argument argument to the forward and inverse projection function and the function return is a type declared (in the header file `projects.h`) as:

```
typedef struct { double x, y; }     XY;
typedef struct { double lam, phi; } LP;
```

which are the respective x and y Cartesian coordinates respective longitude, λ, and latitude, ϕ, geographic coordinates in radians. If either the forward or inverse function fail to perform a conversion, both values in the returned structure are set to `HUGE_VAL` as defined in the `math.h`.

Two additional notes should be made about the header file `projects.h`: it contains includes to the system header files `stdlib.h` and `math.h`, and several predefined constants such as multipliers `DEG_TO_RAD` and `RAD_TO_DEG` to respectively convert degrees to and from radians.

To illustrate usage, the following is an example of a filter procedure designed to convert input pairs of latitude and longitude values in decimal degrees to corresponding Cartesian coordinates using the Polyconic projection with a central meridian of 90°W and the Clarke 1866 ellipsoid:

```
#include <stdio.h>
#include <lib_proj.h>
main(int argc, char **argv) {
    static char *parms[] = {
       "proj=poly",
       "ellps=clrk66",
       "lon_0=90W"
    };
    PJ *ref;
LP idata;
    XY odata;

    if ( ! (ref = pj_init(sizeof(parms)/sizeof(char *), parms)) ) {
       fprintf(stderr, "Projection initialization failed\n");
       exit(1);
    }
    while (scanf("%lf %lf", &idata.phi, &idata.lam) == 2) {
```

```
        idata.phi *= DEG_TO_RAD;
        idata.lam *= DEG_TO_RAD;
        odata = pj_fwd(idata, ref);
        if (odata.x != HUGE_VAL)
           printf("%.3f\t%.3f\n", odata.x, odata.y);
        else
           printf("data conversion error\n");
    }
    exit(0);
}
```

To test the program, the script

```
./a.out <<EOF
0 -90
33 -95
77 -86
EOF
```

should give the results:

```
0.000   0.000
-467100.408     3663659.262
100412.759      8553464.807
```

When executing `pj_init` the projection system allocates memory for the structure pointed to by the return value. This allocation is complex and consists of one or more additional memory allocations to assign substructures referenced within the base structure. In applications where multiple calls are to `pj_init` are made and where the previous initializations are no longer needed it is advisable to free up the memory associated with the no longer needed structures by calling `pj_free`.

In some cases it is convenient to include:

```
#define PROJ_UV_TYPE
```

before the inclusion of the `lib_proj.h` header file. This changes the declaration of the forward and inverse entries to having a

```
typedef struct { double u, v; } UV;
```

type for both the first argument and functional return. The included program `lproj` is an example where this is used and facilitates the processing of the I/O that can be either forward or inverse projection which is performed by substituting the appropriate forward or inverse procedure interchangeably.

1.2 Projection factors.

Various details about a projections behavior including scale factors at selected geographic coordinates can be determined with the function:

```
#include <lib_proj.h>

int pj_factors(LP lp, PJ *P, double h, struct FACTORS *fac);
```

Argument `lp` is the coordinate where the factors are to be determined, `P` points to the projection's control structure, `h` numerical derivative increment and `fac` is a structure defined in `lib_proj.h` as:

```
struct DERIVS {
double x_l, x_p; /* derivatives of x for lambda-phi */
double y_l, y_p; /* derivatives of y for lambda-phi */
};
struct FACTORS {
struct DERIVS der;
double h, k;    /* meridional, parallel scales */
double omega, thetap;   /* angular distortion, theta prime */
double conv;    /* convergence */
double s;       /* areal scale factor */
double a, b;    /* max-min scale error */
int code;       /* info as to analytics, see following */
};
#define IS_ANAL_XL_YL 01   /* derivatives of lon analytic */
#define IS_ANAL_XP_YP 02   /* derivatives of lat analytic */
#define IS_ANAL_HK    04   /* h and k analytic */
#define IS_ANAL_CONV 010   /* convergence analytic */
```

The variable `code` has bits set according to the `define`s where "analytic" refers to equations within the projections providing the values rather than their determination by numerical differentiation.

The argument `h` may be 0. and a suitable default value will be used.

For a more complete, mathematical description of the elements in `FACTORS` see Section 3.8.

1.3 Error handling.

Error detection is a combination of using the C library facilities relating to `error` and the global projlib variable `pj_error`. To simplify matters for the user, the application program only need to sense the `pj_error` for a non-zero value. If the value is greater than zero a C library procedure detected an error and if less than zero a libproj4 procedure detected an error.

To get a string that describes the error use the following:

```
#include lib_proj.h

char *emess;

emess = pj_strerror(pj_errno);
```

A null pointer is returned if `pj_errno==0`.

1.4 Character/Radian Conversion.

Two procedures in the LIBPROJ4 library are provided to perform conversion between human readable character representation of geodetic coordinates and internal floating point binary. These procedures are summarized by the following prototypes:

```
#include <lib_proj.h}

double pj_dmstor(const char * str, char ** str)
char *pj_rtodms(char *str, double rad, const char * signt)
void pj_set_rtodms(int frac, int con_w)
```

The pj_dmstor function is patterned after the C language library strtod function where str is a character string to be read for a DMS value to be returned as the function value and the second character pointer returns a pointer to the next character in the string after the successfully decoded string. If a proper DMS value is not found then a 0 is returned and a HUGE_VAL is returned for bizarre conversion errors. In the latter case pj_errno may be set with a -16 value.

Function rtodms performs output formatting and creates a DSM string from the input rad. The argument signt is a two character string where the first character is to be taken as the positive sign suffix and the second as the negative sign suffix. Normally, signt will either be "NS" or "EW". If signt is 0 then normal numeric minus sign prefixes the numeric output.

Normal output of pj_rtodms formats to 3 decimal digits of seconds but this precision can be adjusted with the pj_set_rtodms function by specifying the number of significant digits to use with frac. If the argument con_w argument is not 0 then constant width values are output (often useful in map labeling or tabular values).

1.5 Limiting Selection of Projections

Many applications will only need a small subset of the projections contained in the library libproj.a, but unless some action is taken, all of the projections will be linked into the final process. This is not a problem unless the memory requirements of the application are to be kept small or access to projections is to be restricted.

If there is a need to limit the number of projections, a simple two-step process needs to followed. First create a header file, my_list.h for example, that contains a list of macro calls PROJ_HEAD(*id*,*text*', one for each projection to be part of the application program. Argument *id* is the acronym of the projection and argument *text* is the ASCII string describing the program (what appears after the colon in **proj**'s -l execution. The header file, nad_list, for program **nad2nad** is a an example:

```
/* projection list for program my_prog */
PROJ_HEAD(lcc, "Lambert Conformal Conic")
PROJ_HEAD(omerc, "Oblique Mercator")
PROJ_HEAD(poly, "Polyconic (American)")
PROJ_HEAD(tmerc, "Transverse Mercator")
PROJ_HEAD(utm, "Universal Transverse Mercator (UTM)")
```

An easy way to create this list is to copy and edit the file pj_list.h in the source distribution, which contains the entire listing of available projections, and edit out of the copy all lines of unwanted projections.

Next, in one of the program code modules that includes the header file projects.h, precede the include statement with:

```
#define PJ_LIST_H "my_list.h"
```

Be careful to only put this include in only one of the code modules because this define action causes the initialization of the global pj_list and multiple initializations will cause havoc with the linker.

Chapter 2

Internal Controls

To discuss the internal control of this system the description will be based upon following the flow of the process from projection initialization to coordinate conversion. Although extracts of the code and data structures will be presented here it may be helpful for the reader to follow the description with frequent references to the source code.

2.1 Initialization Procedures.

To initiate the cartographic transformation system it is necessary to execute a procedure that will decode the user's control input into internally recognized parameters and to establish a myriad of computational constants and process controls and return to the calling procedure a reference to employ when performing transformations. In this system the entry is the procedure `pj_init` is passed a argument count and character array in a manner similar to a C program's `main`. The first operation `pj_init` performs is to put the list of arguments into a linked list described in the next section.

The reason for this copy operation is that it allows the system to add arguments to the list and not violate `const` attributes of the input list and it also allows marking each argument element that is used by the system. This latter feature is useful in giving an audit trail for debugging usage of system.

The first extraction from the input list is to determine the identifier of the projection to be used (`+proj=<id>`) and locating the entry `id` in the list:

```
struct PJ_LIST {
    char *id; /* projection keyword */
    PJ *(*proj)(PJ *); /* projection entry point */
    char  * const *descr; /* description text */
};
```

The following extract from the `lib_proj.h` header file shows how the projection list is declared and initialized:

```
/* Generate pj_list external or make list from include file */
#ifndef PJ_LIST_H
extern struct PJ_LIST pj_list[];
#else
#define PROJ_HEAD(id, name) \
extern PJ *pj_##id(PJ *); extern char * const pj_s_##id;
#define DO_PJ_LIST_ID
#include PJ_LIST_H
```

```
#undef DO_PJ_LIST_ID
#undef PROJ_HEAD
#define PROJ_HEAD(id, name) {#id, pj_##id, &pj_s_##id},
struct PJ_LIST
pj_list[] = {
#include PJ_LIST_H
{0,     0,  0},
};
#undef PROJ_HEAD
#endif
```

In all but one situation of the usage of `lib_proj.h` the identifier `PJ_LIST_H` is undefined and thus only the external declaration of the projection list `pj_list` is made. In the case of the file `pj_list.c` the only code in the file is:

```
#define PJ_LIST_H "pj_list.h"
#include "lib_proj.h"
```

which result in the following actions:

- the `PROJ_HEAD` macro is defined as a declaration of the external projection function and an external description character string,
- the header file `pj_list.h` containing a list of `PROJ_HEAD` statement is read,
- `PROJ_HEAD` is redefined so as to create a structure array and initializes that array by re-reading the header file `pj_list.h`

The reason for this seemingly convoluted operation is to simplify the installation of new projections by merely creating the the `PROJ_HEAD` macro once in the file containing the projection code and then simply copying this line into the list-defining header file.

Once the projection initialization entry is determined from the list the next operation is to call the projection entry defined in the list structure with a zero (null) argument. The projection procedure will return a pointer to the `PJconsts` structure whose top portion is defined in `lib_proj.h`. This structure pointer is what is eventually returned by `pj_init` to the calling program after its contents are fully initialized. The reason for having the projection return the structure pointer is that the complete definition and size is defined by the selected projection.

At this stage all of the elements after the first five of the structure `PJconsts` are filled in by following operations of `pj_init`. These components are found to be commonly used and projection independent and thus more efficiently determined by a common process.

The final step is to re-call the projection entry point previously used but now with the pointer to the `PJconsts` stucture as the argument and allow the projection to complete the initialization of the structure based upon the already initialize elements and other options in the argument link list that are unique to the projection. Note that the base address of the base address of the argument list is now stored in the structure.

If all goes well, the pointer to the structure `PJconsts` is returned to the user as the functional return of `pj_init`.

2.1.1 Setting the Earth's figure.

In initializing the `PJconsts` stucture the elliptical parameters are the first parameters determined by a call to the function `pj_ell_set`. Its first operation is to search the parameter link-list for the definition of `+R=<radius>` and if found, the remainder of the initialization is for a spherical earth regardless of any ellipsoid parameters on the list.

If the radius is not on the list, then a search the argument `+ellps=<id>` and a search of the table

```
struct PJ_ELLPS {
    char  *id;    /* ellipse keyword name */
    char  *major; /* a= value */
    char  *ell;   /* elliptical parameter */
    char  *name;  /* comments */
};
```

is made and if found, the ellipsoid parameters from the second and third character fields are pushed onto the parameter linked list.

The remainder of the `PJconsts` fields related to the ellipsoid or sphere are now determined.

If neither a radius nor ellipsoid constants are found, an error condition exists.

2.2 Determinations from the argument list.

Control options are the list of projection parameters typically obtained from run lines of programs or data bases. They consist of the option name optionally followed by an equal sign and an option value that may be a integer, floating, degree-minute-second (MDS or character string value. Control options may be prefixed with a `+` sign that is ignored by following functions.

2.2.1 Creating the list.

One of the first functions of initialization of projection procedures in LIBPROJ4 is to convert the string array `argv` into a linked list with the structure:

```
struct ARG_list {
    struct ARG_list *next;
    char used;
    char param[1];
};
```

When each control parameter is stored in the list, the flag `used` is set to zero. If the parameter is somehow tested or the argument used the flag is set to one. This serves as an audit trail on projection usage if the verbose diagnostic call is employed.

The argument string is placed into the list with execution of the function:

```
#include <lib_proj.h>

paralist *pj_mkparam(char *str);
```

where `paralist` is a typedef of list structure. If `pj_mkparam` is unable to allocate memory for the new argument then a `NULL` value is returned.

The calling program must use the returned pointer to either establish the starting point of a list or add to the "`next`" value at the end of an existing list.

2.2.2 Using the parameter list

The function `pj_param` provides for searching for parameters and returning their value from `paralist`.

```
#include <lib_proj.h>

PVALUE pj_param(paralist *pl, const char *opt)
```

where

```
typedef union {
    double  f;
    int  i;
    const char *s;
} PVALUE;
\begin{center}
```

Upon calling `pj_param` the argument `opt` character string contains the name of the option desired with a prefix character of how the the option argument is to be treated. The following is a list of the prefix characters and the nature of the return value of `pj_param`.

`t`	test for the presence of the string in the list. Return integer 1 is present else 0.
`i`	treat the option argument as integer and return the binary value.
`d`	treat the option argument as a real number and return double as the result.
`r`	argument is degree-minute-second input and return type double value in radians.
`s`	argument is a character string and return pointer to string.
`b`	argument is boolean; return integer 0 if value "`F`", "`f`", "`0`" or integer 1 if the value is "`T`", "`t`" or "`1`".

In all cases where there is no argument value a 0 or `NULL` value is returned.

In practice, the `b` type is rarely used and it is understood that the presences or absence of the option serves as a boolean flag with the `t` test.

2.3 Computing projection values

A review of the operations that are performed by the entry points `pj_fwd` and `pj_inv` is necessary in order to understand what is performed by the system before calling the individual projection procedures. The following operations are deemed to be common to all forward projections even though they maybe seldom used in some cases:

- The range of the latitude and longitude arguments is check. The absolute value of latitude must be less than or equal to 90° ($\pi/2$ radians) and the absolute value of longitude must be less than or equal to 10 radians (573°).
- Clear error flags.
- If geocentric latitude option is selected the latitude is changed to geodetic latitude.
- Central meridian is subtracted from the longitude.
- If over-ranging is not selected the longitude is reduced to be between ±180°.

- The projection procedure is called.
- It errors, then set x–y to `HUGE_VAL` and return, else x–y values are multiplied by the Earth's radius or major elliptical axis, false Northing and Easting are added and each are scaled to the selected units.

The main thing to note is that the projection functions only deal with longitude reduced to the central meridian (no $\lambda - \lambda_0$ terms) and an unit radius/major-axis Earth.

In the case of the inverse projection, fewer checks of the input data can be done by the inverse projection entry:

- Clear error flags.
- Adjust the Cartesian coordinates by rescaling, subtracting the false Easting and Northing and dividing out the Earth's radius or major-axis.
- Call the inverse projection.
- If errors, set λ–ϕ to `HUGE_VAL` and return.
- Add central meridian to returned longitude.
- If over-ranging not selected reduce longitude range to between
- If geocentric latitude specified, change geodetic latitude to geocentric.

2.4 Projection Procedure.

Because the library was intended to have a large number of projection procedures care was given to facilitating the coding of the procedures and to make them have a similar structure. By following this guideline it is easy to develop new projections (at least as far as the controlling code).

The following is the skeletal outline of a projection procedure:

```
<boiler plate---copyright/disclaimers, etc.>
#define PROJ_PARMS__ \
     <local extensions to PJconsts structure>
#define PJ_LIB__
#include        <lib_proj.h>
PROJ_HEAD(<entry_id>, "<expanded descriptive name>") "\n\t<type>, ...";
<local defines, static variablesi, functions, ...>
FORWARD(<forward_id>);
    <declarations and code for forward>
    xy.x =
    xy.y =
    return (xy);
}
INVERSE(<inverse_id>);
<declarations and code for inverse>
    lp.phi =
    lp.lam =
    return (lp);
}
FREEUP;
    if (P)
        free(P);
```

```
}
ENTRY0(<entry_id>)
    <initialization code>
    P->inv = <inverse_id>;
    P->fwd = <forward_id>;
    ENDENTRY(P)
```

where the material enclosed in angle braces is a form of comment for this demonstration.

The first thing to note is the defining of `PJ_LIB__` which enables sections of the header file that contain definitions and other material unique to the projection procedures. The next item is the definition of `PROJ_PARMS__` that defines extensions to the structure that are unique to the current projection. Looking at the definition in the header file `lib_proj.h`

```
typedef struct PJconsts {
    XY  (*fwd)(LP, struct PJconsts *);
    LP  (*inv)(XY, struct PJconsts *);
    void (*spc)(LP, struct PJconsts *, struct FACTORS *);
    void (*pfree)(struct PJconsts *);
    const char *descr;
    paralist *params;   /* parameter list */
    int over;   /* over-range flag */
    int geoc;   /* geocentric latitude flag */
    double
        a,  /* major axis or radius if es==0 */
        e,  /* eccentricity */
        es, /* e ^ 2 */
        ra, /* 1/A */
        one_es, /* 1 - e^2 */
        rone_es, /* 1/one_es */
        lam0, phi0, /* central longitude, latitude */
        x0, y0, /* easting and northing */
        k0,     /* general scaling factor */
        to_meter, fr_meter; /* cartesian scaling */
#ifdef PROJ_PARMS__
PROJ_PARMS__
#endif /* end of optional extensions */
} PJ;
```

shows how the projection unique values are treated. In cases of very simple projections, the definition may be omitted. Finally the inclusion of the `lib_proj.h` header file.

The `PROJ_HEAD` macro is used to define the entry point to the projection, an expanded description string and a string containing expanded information. The first argument `<entry_id>` **must** match the name used in the `ENTRY0` macro. This identifier argument is prefixed with `PJ_` and is used as the external reference for the projection and is the point where the projection is called for initialization.

There may be more than one entry point and thus more than one `PROJ_HEAD` and `ENTRY0` combinations. A good example of this is the Transverse Mercator projection which has two entries: `tmerc` and `UTM`. The Universal Transverse Mercator is a usage of the Transverse Mercator with added constraints and controls of parameters but remaining computations are identical.

Additional variants of `ENTRY0(<id>` are `ENTRYn,<id>,<args>` where n is 1 or 2 and which have a corresponding number of identifier `args` in the macro. The identifiers must be contained in the `PJ_consts` structure as pointers that are to be set to 0 (NULL) at the beginning of initialization.

In all entry cases, the `ENTRY` macros checks the non-null status of the input argument pointing to the structure and if null allocates memory for the structure `PJ_consts` and clears or sets the first five members of the structure and returns with with the structure address. For a non-null input argument control is passed to the following code which should conclude with the macro `ENDENTRY(<arg>)`. In most cases `arg` is the pointer to the structure `PJconsts` but it can be a call to an static, local function that also returns the pointer.

The `FORWARD` and `INVERSE` macros define the local, static entry points for the respective forward and inverse projection calculations and their addresses are stored in the `PJconsts` structure. In many cases there are two forward and inverse entries for the cases of elliptical and spherical earth and the initializing entry will select the ones to be stored on the basis of non-zero e previously set in `PJconsts`. Occasionally there is only a forward projection for the spherical case and thus only a `FORWARD` section. These two macros also declare the arguments and return structures `xy` and `lp`.

In all cases, including initialization, the identifier pointing to `PJ_consts` is `P`.

Error conditions are best handled by four macros:

- `F_ERROR` for use in forward projection code and sets the global `pj_errno` to -20 and returns,
- `I_ERROR` is the same as above but for inverse projection code,
- `E_ERROR_0` for use in initialization code and it free allocated `PJ_consts` memory and returns a null pointer. It is assumed that some procedure call by the initializing code has already set `pj_errno`.
- `E_ERROR(<no>)` same as above but also sets the external `pj_errno` to the negative argument value.

The complexity of the entry to free the memory allocated to the structure `PJ_consts` is dependent upon how many additional sub allocations have been made. For projections of the spherical Earth there are usually no sub-allocations and the prototype listed earlier is complete. Additional memory sub-allocations to be released is the same as the number of arguments in the initialization entry macros.

2.5 Setting new error numbers.

When developing new procedures or projections for the libproj library where error detection is part of the code do the following steps. Check the program file `pj_strerrno.c` which contains a listing of all the libproj4 error numbers. If a current error condition applies to the new error condition, then use that negative number as the value to be assigned to `pj_errno`. Otherwise, install a new descriptive string at the next to last line of the list `pj_err_list` with a new, negative error number.

Chapter 3

Analytic Support Functions

The material in this chapter expands upon equations and procedures employed by the projection functions and how they are implemented in the C programming environment. In most cases a description of the originating mathematical function is presented rather than just the series or other simplification used for evaluation. The reason for this is that the reader may have insights into how to improve the evaluation and further enhance the performance of the system.

In many cases function naming goes back to early FORTRAN versions of GCTP where an effort was made to collect common computing operations into globally available subroutines. As with projection descriptions, all procedures that deal with ellipsoidal or spherical operations are performed for the unit ellipsoid ($a = 1$) or unit sphere ($R = 1$).

3.1 Ellipsoid definitions

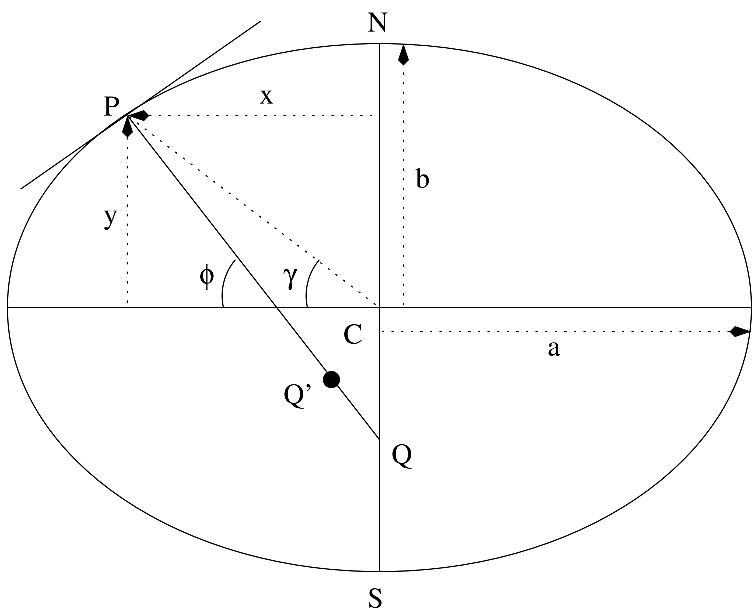

Figure 3.1: The meridional ellipse.

From Fig. 3.1 the components and symbols used in this document for defining the ellipsoid are summarized as follows:

$$\begin{array}{rrcl}
\text{semimajor axis} & a & & \\
\text{semiminor axis} & b & & \\
\text{excenticity} & e^2 & = & \dfrac{a^2-b^2}{a^2} \\
 & & = & \dfrac{e'^2}{1+e'^2} \\
 & & = & 2f - f^2 \\
\text{second excentricity} & e'^2 & = & \dfrac{a^2-b^2}{b^2} \\
 & & = & \dfrac{e^2}{1-e^2} \\
\text{flattening} & f & = & \dfrac{a-b}{a}
\end{array}$$

The angle ϕ is geographic or geodetic latitude and λ is geodetic longitude (the angle of rotation of the meridianal plane about the N-S axis). Geocentric latitude, γ, is infrequently used in projection applications.

The distances PQ' and PQ are the respective radii of the ellipsoid surface in the plane of the meridianal ellipse and normal to the plane of the meridianal ellipse.

$$\text{PQ'} \quad = R \quad = \quad \frac{a(1-e^2)}{(1-e^2\sin^2\phi)^{3/2}} \tag{3.1}$$

$$\text{PQ} \quad = N \quad = \quad \frac{a}{(1-e^2\sin^2\phi)^{1/2}} \tag{3.2}$$

3.2 Meridian Distance—`pj_mdist.c`

A common function among cartographic projections for the ellipsoidal earth is to determine the distance along a meridian from the equator to latitude ϕ. The definition of this distance is the integral of the radius of the spheroid in the plane of the meridian (equation 3.1)

$$M(\phi) = a(1-e^2)\int_0^{\phi} \frac{d\phi}{(1-e^2\sin^2\phi)^{3/2}} \tag{3.3}$$

which can be computed as

$$M(\phi) = a\left(E(\phi,e) - \frac{e^2\sin\phi\cos\phi}{\sqrt{1-e^2\sin^2\phi}}\right) \tag{3.4}$$

where $E(\phi, e)$ is the elliptic integral of the second kind. When e is small (as in the case of the Earth's eccentricity) a means of evaluating the elliptic integral is as follows:

$$\begin{aligned}
E(\phi,e) &= E\phi + \sin\phi\cos\phi(b_0 + \frac{2}{3}b_1\sin^2\phi + \frac{2\cdot 4}{3\cdot 5}b_2\sin^4\phi + \cdots) \\
b_0 &= 1 - E \\
b_n &= b_{n-1} - \left[\frac{(2n-1)!!}{2^n n!}\right]^2 \frac{e^{2n}}{2n-1} \\
E &= 1 - \frac{1}{2^2}e^2 - \frac{1^2\cdot 3}{2^2\cdot 4^2}e^4 - \cdots - \left[\frac{(2n-1)!!}{2^n n!}\right]^2 \frac{e^{2n}}{2n-1}
\end{aligned}$$

In the LIBPROJ4 library three functional entries are used in the meridional distance calculations:

```
void *pj_mdist_ini(double es)
double pj_mdist(double phi, double sphi, double cphi, const void *en);
double pj_inv_mdist(double dist, const void *en)
```

Function `pj_mdist_ini` determines E and the series coefficients b_n for the specified eccentricity argument (e^2) and returns a pointer to a structure of these values, `en`, for use by the forward and inverse functions. In the case of an unreasonably large value of e^2, function `pj_mdist_ini` could fail and thus return a null pointer. The degree required by the series is automatically determined by the procedure so as to ensure precision commensurate with the type `double` on the host hardware.

Function `pj_mdist` returns the distance from the equator to the latitude `phi`. In the interests of avoiding repeated evaluation of sine (`sphi`) and cosine (`cphi`) of latitude (almost always computed for other reasons in the calling procedures) these values are included in the argument list. Function `pj_inv_mdist` returns the latitude for a distance `dist` from the equator. In both the forward and inverse case the sign of the latitude and distance is carried though the evaluation so that a negative latitude gives a negative meridian distance and conversely.

3.2.1 Rectifying Latitude

The rectifying latitude, μ (or ω) is a latitude on a sphere determined by the ratio of the distance from the equator for a point on the ellipsoid at latitude ϕ divided by the distance over the ellipsoid from the equator to the pole:

$$\mu = \frac{\pi}{2} \cdot \frac{M(\phi)}{M(\pi/2)} \tag{3.5}$$

where the function M is the meridian distance from (3.4).

3.3 Conformal Sphere—`pj_gauss.c`

Determinations of oblique projections on an ellipsoid can be difficult to solve and result in long, complex computations. Because conformal transformations can be made multiple time without loss of the conformal property a method of determining oblique projections involves conformal transformation of the elliptical coordinates to coordinates on a conformal sphere. The transformed coordinates can now be translated/rotated on the sphere and then converted to planar coordinates with a conformal spherical projection. Pearson [10] gives a development of the conformal transformation but assumes a zero constant of integration.

The conformal transformation of ellipsoid coordinates (ϕ, λ) to conformal sphere coordinates (χ, λ_c) is

$$\chi = 2\arctan\left[K \tan^C(\pi/4 + \phi/2)\left(\frac{1 - e\sin\phi}{1 + e\sin\phi}\right)^{Ce/2}\right] - \pi/2 \tag{3.6}$$

$$\lambda_c = C\lambda \tag{3.7}$$

$$R_c = \frac{\sqrt{1 - e^2}}{1 - e^2\sin^2\phi_0} \tag{3.8}$$

where λ is relative to the longitude of projection origin, R_c is radius of the conformal sphere and

$$C = \sqrt{1+\frac{e^2 cos^4\phi_0}{1-e^2}} \tag{3.9}$$

$$\chi_0 = \arcsin\left(\frac{\sin\phi_0}{C}\right) \tag{3.10}$$

$$K = \tan(\chi_0/2+\pi/4)/\left[\tan^C(\phi_0/2+\pi/4)\left(\frac{1-e\sin\phi_0}{1+e\sin\phi_0}\right)^{Ce/2}\right] \tag{3.11}$$

where χ_0 is the latitude on the conformal sphere at the central geographic latitude of the projection.

To determine the inverse solution, geographic coordinates from Gaussian sphere coordinates, execute:

$$\lambda = \lambda_c/C \tag{3.12}$$

$$\phi = 2\arctan\left[\frac{\tan^{1/C}(\chi/2+\pi/4)}{K^{1/C}\left(\frac{1-e\sin\phi_{i-1}}{1+e\sin\phi_{i-1}}\right)^{e/2}}\right]-\pi/2 \tag{3.13}$$

with the initial value of $\phi_{i-1}=\chi$ and ϕ_{i-1} iteratively replaced by ϕ until $|\phi-\phi_{i-1}|$ is less than an acceptable error value.

Procedures to compute the transformation are:

```
#include <lib_proj.h>

void *pj_gauss_ini(double es, double phi0,
           double *chi0, double *rc)
LP pj_gauss(LP arg, const void *en)
LP pj_gauss_inv(LP arg, const void *en)
```

The initialization procedure `pj_gauss_ini` returns a pointer to a control array for forward and inverse conversion at the latitude of origin `phi0` (ϕ_0). It also returns the radius of the Gaussian sphere (`rc`). Procedures `pj_gauss` and `pj_gauss_inv` are respective forward and inverse conversion of the latitude and longitude to and from the Gaussian sphere. The storage pointed to by `en` should be release back to the system upon completion of conversion usage.

3.3.1 Simplified Form of Conformal Latitude.

A common determination of the conformal latitude is made by setting $K=1$ (based upon zero constant of integration which causes $\chi\to 0$ as $\phi\to 0$) and set $C=1$ which seems to be equivalent to similar to having $\chi\to\pi/2$ as $\phi_0\to\pi/2$. Equation 3.6 now becomes:

$$\chi = 2\arctan\left[\tan(\pi/4+\phi/2)\left(\frac{1-e\sin\phi}{1+e\sin\phi}\right)^{e/2}\right]-\pi/2 \tag{3.14}$$

$$\lambda_c = \lambda \tag{3.15}$$

Determining ϕ from χ is the same as discussed for equation 3.13.

The radius of the conformal sphere is determined by:

$$R_c = \frac{\cos\phi_0}{\cos\chi_0}(1-e^2\sin^2\phi_0)^{-1/2} \tag{3.16}$$

This new sphere radius is not how it is phrased by Snyder [14, page 160] or Thomas [18, page 134] but it serves as a useful equivalence when making a replacement funtion for `pj_gauss_ini`. The derivation of this factor was based upon the requirement of unity scale factor at the Stereographic projection origin. For the moment, this is the only projection that employs this procedure so beware in applying it in other cases.

Although the precedure to perform the simplified Gauss latitude need not be as complex, the operations are made compatible with the general use for compatibility.

```
#include <lib_proj.h>

void *pj_sgauss_ini(double es, double phi0,
                    double *chi0, double *rc)
LP pj_sgauss(LP arg, double *en)
LP pj_sgauss_inv(LP arg, double *en)
```

3.4 Authalic Sphere—pj_auth.c

Authalic operations relate to the sphere having the same surface area of an elliptical earth. From the integral definition:

$$\int R^2 \cos\beta \, d\beta \quad = \quad a^2(1-e^2)\int \frac{\cos\phi}{(1.-e^2\sin^2\phi)^2} d\phi \tag{3.17}$$

which is readily solved by binomial expansion of the denominator and term-by-term integration:

$$\begin{aligned} R^2 \sin\beta &= a^2(1-e^2)\sin\phi\left(1+\frac{2}{3}e^2\sin^2\phi+\frac{3}{5}e^4\sin^4\phi+\frac{4}{7}e^6\sin^6\cdots\right) \\ &= a^2(1-e^2)\sin\phi\sum_{n=0}\frac{1+n}{1+2n}e^{2n}\sin^{2n}\phi \end{aligned} \tag{3.18}$$

The constants of integration are eliminated to main equality when $\phi = \beta = 0$ and R (radius of the authalic sphere) is determined by ensuring $\phi = \beta = \pi/2$ and thus is obtained from:

$$R^2 \quad = \quad a^2(1-e^2)\sum_{n=0}\frac{1+n}{1+2n}e^{2n} \tag{3.19}$$

Finally, the authalic latitude is:

$$\beta \quad = \quad \arcsin\left(\frac{\sin\phi\displaystyle\sum_{n=0}\frac{1+n}{1+2n}e^{2n}\sin^{2n}\phi}{\displaystyle\sum_{n=0}\frac{1+n}{1+2n}e^{2n}}\right) \tag{3.20}$$

$$\quad = \quad \arcsin\left(\sin\phi\sum_{n=0}c_{2n}\sin^{2n}\phi\right) \tag{3.21}$$

where c_{2n} are the collapsed constants determined by the initializing process specifying e.

To obtain the geodetic latitude from the authalic latitude the Newton-Raphson process can be used where the initial value of $\phi = \beta$:

$$\phi_+ \quad = \quad \phi + \frac{\sin\beta - \sin\phi\displaystyle\sum_{n=0}c_{2n}\sin^{2n}\phi}{\cos\phi\displaystyle\sum_{n=0}\frac{c_{2n}}{2n+1}\sin^{2n}\phi} \tag{3.22}$$

Another authalic factor (currently lacking a name) is the q function typically defined as:

$$
\begin{aligned}
q &= (1-e^2)\left[\frac{\sin\phi}{1-e^2\sin^2\phi} - \frac{1}{2e}\ln\left(\frac{1-e\sin\phi}{1+e\sin\phi}\right)\right] \qquad (3.23)\\
&= 2(1-e^2)\sin\phi\sum_{n=0}\frac{1+n}{1+2n}e^{2n}\sin^{2n}\phi\\
&= \frac{R^2}{2a^2}\sin\beta
\end{aligned}
$$

The series form of the function is used in the library function `qsfn`.

LIBPROJ4 entries:

```
#include <lib_proj.h>

void* pj_auth_ini(double es, double *r)
double pj_qsfn(double phi, void*i en)
double pj_auth_lat(double phi, void* en)
double pj_auth_inv(double beta, void* en)
```

3.5 Axis Translation—pj_translate.c

This set of procedures performs axis translations for the spherical coordinate system. The elliptical system can only be translated about the polar axis— a process performed by the λ_0 or central meridian factor. One way for elliptical projections to perform general translation is transformation of the elliptical coordinates to the sphere and subsequent use of this procedure.

Mathematically, the forward translation is performed by:

$$\sin(\phi') = \sin\alpha\sin\phi - cos\alpha\cos\phi\cos\lambda \qquad (3.24)$$

$$\tan(\lambda'-\beta) = \frac{\cos\phi\sin\lambda}{\sin\alpha\cos\phi\cos\lambda + \cos\alpha\sin\phi} \qquad (3.25)$$

and the inverse translation performed by:

$$\sin(\phi) = \sin\alpha\sin\phi' + cos\alpha\cos\phi'\cos(\lambda'-\beta) \qquad (3.26)$$

$$\tan\lambda = \frac{\cos\phi'\sin(\lambda'-\beta)}{\sin\alpha\cos\phi'\cos(\lambda'-\beta+\cos\alpha\sin\phi'} \qquad (3.27)$$

The latitude α is the position of the North Pole of the original coordinates system on the new system at a longitude β east of the central meridian of the new coordinates ($\lambda' = 0$). In most applications $\beta = 0$.

The library translation functions are:

```
#include <proj_lib.h>

LP pj_translate(LP base, void *en);
LP pj_inv_translate(LP shift, void *en);
void *pj_translate_ini(double alpha, double beta);
```

Execution of the initializing function `pj_translate_ini` will return a pointer to a structure containing constants for the forward and inverse operations. A `NULL` value will be returned if the procedure failed to successfully obtain memory.

Function `pj_translate` returns the translated original coordinates and conversely, `pj_translate` returns the translated coordinates back to the original values. Users must execute `free(en)` upon end of usage.

3.6 Transcendental Functions—pj_trans.c

In order to avoid domain errors in calling several of the standard C library functions several alternate entries are used:

```
#include <lib_proj.h>

double pj_asin(double)
double pj_acos(double)
double pj_sqrt(double)
double pj_atan2(double, double)
```

The `pj_asin` and `pj_acos` check that arguments whose absolute value exceeds unity by a small amount are successfully resolved. Similarly a sufficiently small negative argument to `pj_sqrt` will cause a return of zero. If both the arguments to `pj_atan2` are sufficiently small it will return a zero value.

3.7 Miscellaneous Functions

These are short functions that date from origins in the GCTP system and perform evaluations for various projections. Part of the purpose of developing GCTP was to minimize repetitive program code.

3.7.1 Isometric Latitude kernel.

The function t

$$t = \tan(\pi/4 + \phi/2)\left(\frac{1 - e\sin\phi}{1 + e\sin\phi}\right)^{e/2} \tag{3.28}$$

is the kernel of $\ln(t)$ (Isometric latitude) that performs conformal mapping of a spheroid to the plane. The kernel is kept separate because it is also frequently used in the inverse form where t^e is evaluated.

```
#include <lib_proj.h>

double pj_tsfn(double phi, double sinphi, e);
```

3.7.2 Inverse of Isometric Latitude.

This function determines the geodetic latitude from the isometric latitude $\tau = \ln(t)$. The procedure is to iteratively solve for ϕ_+ until a sufficiently small difference between evaluations occurs.

$$\phi_+ \quad = \quad \pi/2 - 2\arctan\left[t\left(\frac{1 - e\sin\phi}{1 + e\sin\phi}\right)^{e/2}\right] \tag{3.29}$$

where

$$t \quad = \quad \exp(-\tau)$$

and using an initial value of:

$$\phi \quad = \quad \pi/2 - 2\arctan(t)$$

Library function prototype:

```
#include <lib_proj.h>

double pj_phi2(double tau, double e);
```

It is unknown how the library function got its name.

3.7.3 Parallel Radius.

The distance of a point at latitude ϕ from the polar axis. Also termed the radius of a parallel of latitude (distance X in figure 3.1 and equation 3.2).

$$m = N\cos\phi = \frac{a\cos\phi}{\sqrt{1-e^2\sin^2\phi}} \tag{3.30}$$

where N is the radius of curvature of the ellipse perpendicular to the plane of the meridian. A unit major axis (a) is used. The LIBPROJ4 prototype is:

```
#include <lib_proj.h>

double pj_msfn(double sinphi, double cosphi, es);
```

3.8 Projection factors.

The meaning of *factors* here is the definition of how a projection performs in terms of various distortions and scaling errors. In some cases analytic functions are readily available that can be included within the individual projections files and available through the `PJconsts` structure. However, it is felt that a numeric determination of these factors is preferable because they are an independent evaluation that determines the factors by execution of the projection code and thus perform a check on these implementations and not upon the merely the evaluation of the factor procedure.

3.8.1 Scale factors.

Two important factors about a projection are the scaling at a given geographic coordinate which is defined by:

$$h = \left[\left(\frac{\partial x}{\partial\phi}\right)^2 + \left(\frac{\partial y}{\partial\phi}\right)^2\right]^{1/2} / R \tag{3.31}$$

$$k = \left[\left(\frac{\partial x}{\partial\lambda}\right)^2 + \left(\frac{\partial y}{\partial\lambda}\right)^2\right]^{1/2} / m \tag{3.32}$$

$$R = \frac{a(1-e^2)}{(1-e^2\sin^2\phi)^{3/2}} \tag{3.33}$$

where h and k are the scale factors along the respective meridian and parallel. R is the ellipsoid radius in the plane of the meridian and m is the parallel radius [3.30]. These equations are for the ellipsoidal Earth but can be readily simplified for the spherical case by setting $e = 0$. Respective scale error is computed from the h and k factors by subtracting 1.

Additional factors to be computed are:

$$a' = (h^2 + k^2 + 2hk\sin\theta')^{1/2} \tag{3.34}$$

$$b' = (h^2 + k^2 - 2hk\sin\theta')^{1/2} \tag{3.35}$$

where

$$\sin\theta' = \frac{\dfrac{\partial y}{\partial\phi}\dfrac{\partial x}{\partial\lambda} - \dfrac{\partial x}{\partial\phi}\dfrac{\partial y}{\partial\lambda}}{\dfrac{a^2(1-e^2)hk\cos\phi}{(1-e^2\sin^2\phi)^2}} \tag{3.36}$$

From a' and b' the respective maximum and maximum scale factors are obtained from

$$a = \frac{a'+b'}{2} \tag{3.37}$$

$$b = \frac{a'-b'}{2} \tag{3.38}$$

and the area scale factor found from

$$S = hk\sin\theta' \tag{3.39}$$

In the case of conformal projections the scale factors must be equal and thus the angular distortion give by

$$\omega = \arcsin\left(\frac{|h-k|}{h+k}\right) \tag{3.40}$$

will be zero.

The remaining element of the projection factors is *convergence* or *grid declination* which is the azimuth of grid north (x or Northing axis) in relation to true north. It is determined by:

$$\gamma = \arctan 2\left(\frac{\frac{\partial y}{\partial \lambda}}{\frac{\partial x}{\partial \lambda}}\right) \tag{3.41}$$

Normally only of interest in formal military or cadastral grid systems.

When the projection modules are not able to provide the values for the partial derivatives then the following numeric method is used:

$$\frac{\partial f_{0,0}}{\partial z} = \frac{1}{4\delta}(f_{1,1} - f_{-1,1} + f_{1,-1} - f_{-1,-1})O(\delta^2) \tag{3.42}$$

The function f is the forward projection used in the procedure `pj_deriv` which calculates the Cartesian coordinates for the four δ offsets from the central point and computes the four partial derivatives. Note that this method may fail if the central point is within δ of the limits of the projection.

Chapter 4

Cylindrical Projections.

The mathematical characteristics of normal cylindical projections is of the form:

$$x = f(\lambda) \tag{4.1}$$
$$y = g(\phi) \tag{4.2}$$

That is, both lines of constant parallels and meridians are straight lines. The term *normal cylindrical* is used here to denote the usage where the axis of the cylinder is coincident with the polar axis. In the transverse and oblique cylindricals the parallels and meridians are complex curves.

Although the example figures of the cylindrical projections are of the entire Earth the cylindrical projection is poorly suited for very small scale mapping because of distortion of the polar regions. However, large scale usage of Mercator in all normal, transverse and oblique forms is in common usage in regions bordering the cylinder's tangency or secant lines. The normal Mercator projection is also in common use in navigation because of the property of a loxodrome being a straight line.

4.1 Normal Aspects.

4.1.1 Arden-Close.

`+proj=ardn_cls` (Fig. 4.1 Mean of Mercator and Cylindrical Equal-Area projections.

$$y_1 = \ln\tan\left(\frac{\pi}{4} + \frac{\phi}{2}\right) \qquad y_2 = \sin\phi \tag{4.3}$$
$$x = \lambda \qquad y = (y_1 + y_2)/2 \tag{4.4}$$

4.1.2 Braun's Second (Perspective).

`+proj=braun2` Fig. 4.1 Ref. [15, p. 111]

$$x = \lambda \qquad y = \frac{7}{5}\sin\phi/\left(\frac{2}{5} + \cos\phi\right) \tag{4.5}$$

4.1.3 Cylindrical Equal-Area.

`+proj=cea [+lat_0= | +lat_ts=]` Fig. 4.1

Standard parallels (0° when omitted) may be specified that determine latitude of

Table 4.1: Alternate names for the Cylindrical Equal-Area projection and their associated control option.

Projection Name	(`lat_ts=`) ϕ_0
Lambert's Cylindrical Equal-Area	$0°$
Berhrmann's Projection (1910)	$0°$
Limiting case of Craster	$37°4'$
Trystan Edwards	$37°24'$
Gall's Orthographic, Peter's	$45°$
Peter's Projection	$44.138°$ (Voxland)
	$46°2'$ (Maling)
M. Balthasart's Projection	$55°$ (Snyder)
	$50°$ (Maling)

true scale ($k = h = 1$). See Table 4.2 for other names associated with this projection.

$$x = \lambda \cos \phi_0 \qquad y = \frac{\sin \phi}{\cos \phi_0} \tag{4.6}$$

4.1.4 Central Cylindrical.

`+proj=cc` Fig. 4.1 Ref. ([15, p. 107,]
Cylindrical version of the Gnomonic Projection. Of little practical value.

$$x = \lambda \qquad y = \tan \phi \tag{4.7}$$

The transverse aspect by Wetch is given as:

$$x = \frac{\cos \phi \sin \lambda}{1 - \cos^2 \phi \sin^2 \lambda)^{-1/2}} \qquad y = \arctan \left(\frac{\tan \phi}{\cos \lambda} \right) \tag{4.8}$$

4.1.5 Cylindrical Equidistant.

`+proj=eqc [+lat_0= | +lat_ts=]` Fig. 4.3
The simplist of all projections. Standard parallels (0° when omitted) may be specified that determine latitude of true scale ($k = h = 1$). See Table 4.2 for other names associated with this projection and corresponding ϕ_{ts} setting.

$$x = \lambda \cos \phi_{ts} \qquad y = \phi \tag{4.9}$$

4.1.6 Cylindrical Stereographic.

`+proj=cyl_stere [+lat_0=]` Fig. 4.2
Standard parallels (0° when omitted) may be specified that determine latitude of

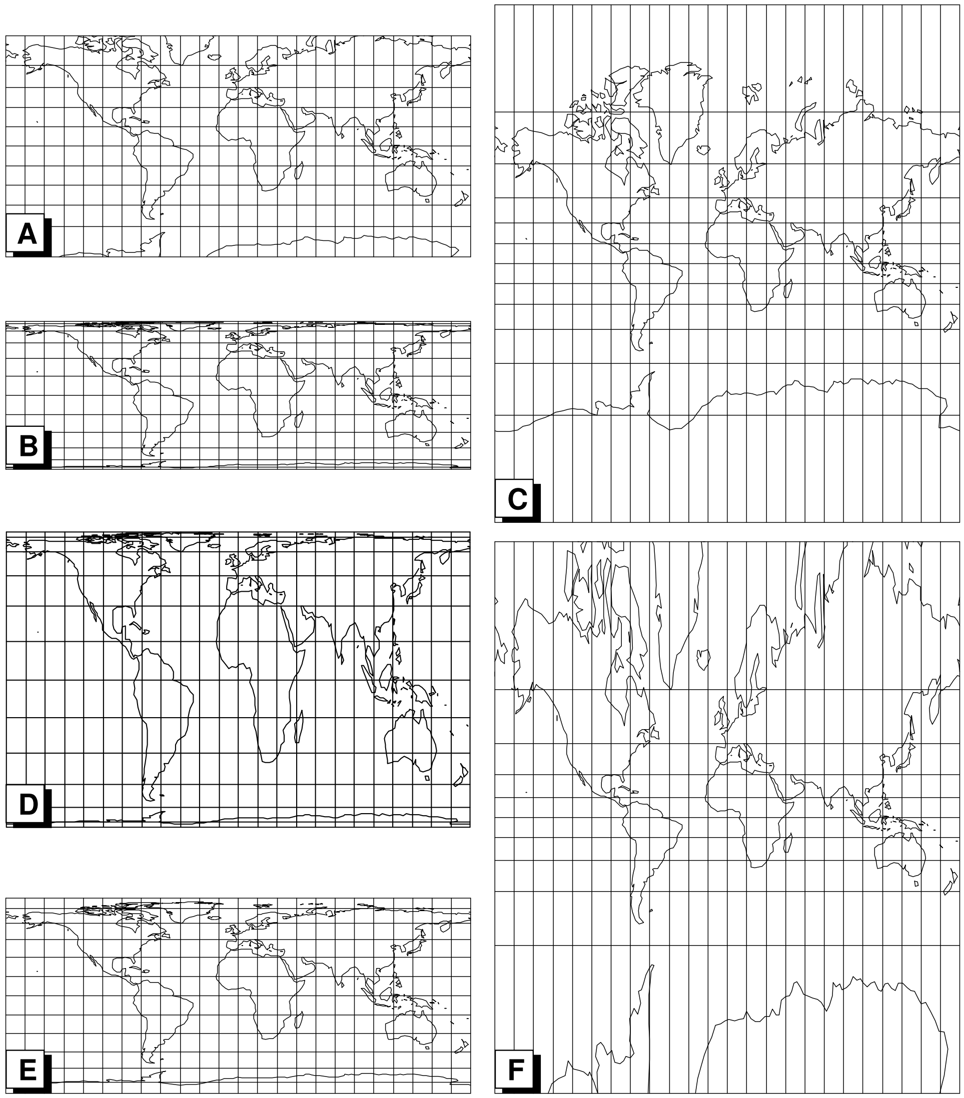

Figure 4.1: Cylinder projections I
A–Arden-Close, **B**–Cylindrical Equal-Area, **C**–Braun's Second, **D**–Gall's Orthograph/Peter's ($\phi_0 = 45°$), **E**–Pavlov and **F**–Central Cylindrical.

true scale ($k = h = 1$). See Table 4.3 for other names associated with this projection.

$$x = \lambda \cos \phi_0 \qquad y = (1 + \cos \phi_0) \tan \frac{\phi}{2} \tag{4.10}$$

4.1.7 Kharchenko-Shabanova.

`+proj=kh_sh` Fig. 4.2

$$x = \lambda \cos \frac{10\pi}{180} \tag{4.11}$$

$$y = \phi(0.99 + \phi^2(0.0026263 + \phi^2 0.10734)) \tag{4.12}$$

Table 4.2: Alternate names for the Equidistant Cylindrical projection and their associated control option.

Projection Name	(`lat_ts=`) ϕ_0
Plain/Plane Chart	$0°$
Simple Cylindrical	$0°$
Plate Carrée	$0°$
Ronald Miller—minimum overall scale distortion	$37°30'$
E. Grafarend and A. Niermann	$42°$
Ronald Miller—minimum continental scale distortion	$43°30'$
Gall Isographic	$45°$
Ronald Miller Equirectagular	$50°30'$
E. Gradarend and A. Niermann minimum linear distortion	$61.7°$

Table 4.3: Alternate names for the Cylindrical Stereographic projection and their associated control option.

Projection Name	(`lat_0=`) ϕ_0
Braun's Cylindrical	$0°$
BSAM or Kamenetskiy's Second	$30°$
Gall's Stereographic	$45°$
Kamenetskiy's First Projection	$55°$
O.M. Miller's Modified Gall	$\frac{2}{\sqrt{3}} = 66.159467°$

4.1.8 Mercator.

`+proj=merc [lat_ts=]` Fig. 4.2 Ref. [14, p. 41, 44]
Scaling may be specified by either the latitude of true scale (ϕ_{ts}) or setting k_0 with `+k=` or `+k_0=`.

Spherical form.

Forward projection:

$$x = k_0\lambda \qquad y = k_0 \begin{cases} \ln\tan\left(\frac{\pi}{4} + \frac{\phi}{2}\right) \\ \frac{1}{2}\ln\left(\dfrac{1+\sin\phi}{1-\sin\phi}\right) \end{cases} \tag{4.13}$$

$$k_0 = \cos\phi_{ts} \tag{4.14}$$

Inverse projection:

$$\lambda = x/k_0 \qquad \phi = \begin{cases} \arctan[\sinh(y/k_0)] \\ \pi - 2\arctan[\exp(-y/k_0)] \end{cases} \tag{4.15}$$

Elliptical form.

Forward projection:

$$x = k_0\lambda \qquad y = k_0 \ln t(\phi) \tag{4.16}$$

$$k_0 = m(\phi_{ts}) \tag{4.17}$$

where $t()$ is the Isometric Latitude kernel function (see 3.7.1 and $m(\phi)$ is the parallel radius at latitude ϕ (see 3.7.3). Inverse projection:

$$\lambda = x/k_0 \qquad \phi = t^{-1}(\exp(-y/k_0)) \tag{4.18}$$

4.1.9 O.M. Miller.

`+proj=mill` Fig. 4.3 Ref. [14, p. 88]

$$x = \lambda \qquad y = \begin{cases} \frac{5}{4}\ln\tan\left(\frac{\pi}{4} + \frac{2}{5}\phi\right) \\ \frac{5}{4}\text{arcsinh}[\tan(\frac{4}{5}\phi)] \\ \frac{5}{8}\ln\left(\dfrac{1+\sin\frac{4}{5}\phi}{1-\sin\frac{4}{5}\phi}\right) \end{cases} \tag{4.19}$$

For the inverse

$$\lambda = x \qquad \phi = \begin{cases} \frac{5}{2}\arctan[\exp(\frac{4}{5}y)] - \frac{5}{8}\pi \\ \frac{5}{4}\arctan[\sinh(\frac{4}{5}y)] \end{cases} \tag{4.20}$$

4.1.10 O.M. Miller 2.

`+proj=mill_2` Fig. 4.3

$$x = \lambda \qquad y = \frac{3}{2}\ln\tan\left(\frac{\pi}{4} + \frac{\phi}{3}\right) \tag{4.21}$$

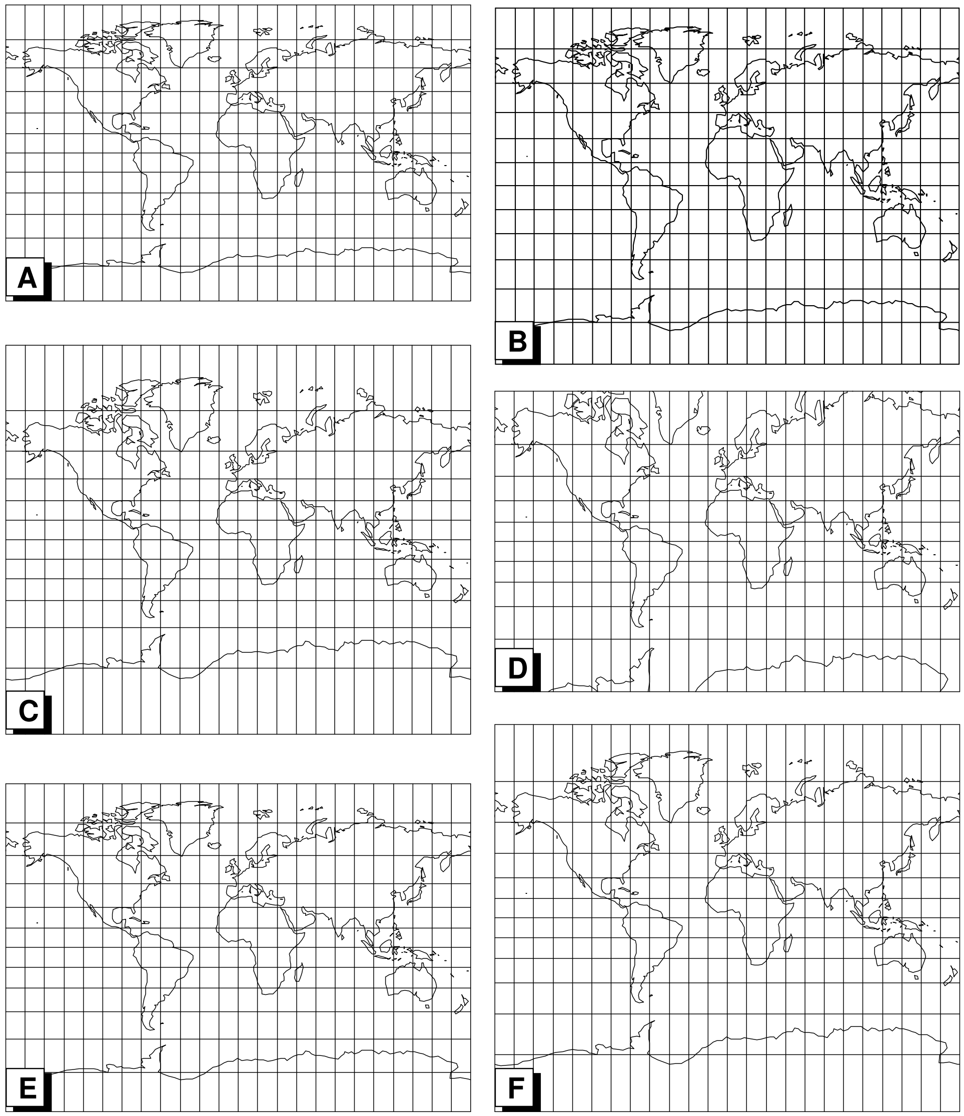

Figure 4.2: Cylinder projections II
A–Cylindrical Stereographc (Braun's), **B**–Gall's Stereographic ($\phi_0 = 45°$), **C**–Kharchenko-Shabanova, **D**–Mercator, **E**–Tobler alternate #2 and **F**–Tobler alternate #1.

4.1.11 Miller's Perspective Compromise.

`+proj=mill_per` Fig. 4.3

$$x = \lambda \qquad y = \left(\sin\frac{\phi}{2} + \tan\frac{\phi}{2}\right) \tag{4.22}$$

4.1.12 Pavlov.

`+proj=pav_cyl` Fig. 4.1

$$x = \lambda \qquad y = \left(\phi - \frac{0.1531}{3}\phi^3 - \frac{0.0267}{5}\phi^5\right) \tag{4.23}$$

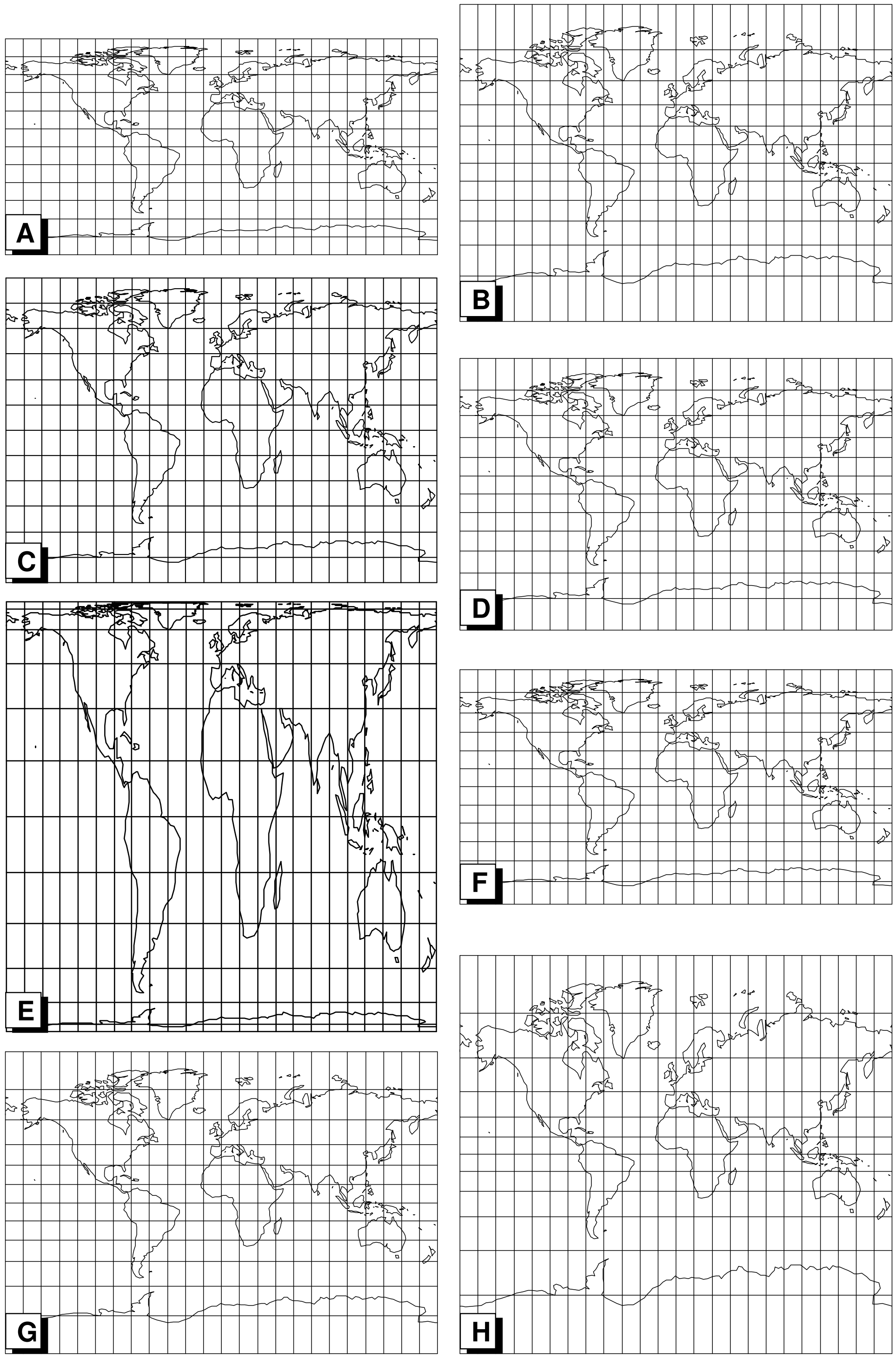

Figure 4.3: Cylinder projections III
A–Cylindrical Equidistant, **B**–Miller, **C**–Gall' Isographic, **D**–Miller 2, **E**–Tobler World in a Square, **F**–Miller Perspective, **G**–Urmayev II, **H**–Urmayev III.

4.1.13 Tobler's Alternate #1

`+proj=tobler_1` Fig. ??
This is alternate to to O.M. Miller's projection.

$$x = \lambda \qquad y = \left(\phi + \frac{1}{6}\phi^3\right) \tag{4.24}$$

4.1.14 Tobler's Alternate #2

`+proj=tobler_2` Fig. ??
This is alternate to to O.M. Miller's projection.

$$x = \lambda \qquad y = \left(\phi + \frac{1}{6}\phi^3 + \frac{1}{24}\phi^6\right) \tag{4.25}$$

4.1.15 Tobler's World in a Square.

`+proj=tob_sqr` Fig. 4.3

$$x = \lambda/\sqrt{\pi} \qquad y = \sqrt{\pi}\sin\phi \tag{4.26}$$

4.1.16 Urmayev Cylindrical II.

`+proj=urm_2` Fig. 4.3

$$\Phi = \left(\frac{\phi^\circ}{10}\right)^2 \tag{4.27}$$

$$x = \lambda \tag{4.28}$$

$$y = \phi\left(1 + \frac{188}{48384}\Phi + \frac{1}{80640}\Phi^2\right) \tag{4.29}$$

The y-axis may be also expressed by: (4.30)

$$c_3 = 0.1275561329783 \tag{4.31}$$

$$c_5 = 0.0133641090422587 \tag{4.32}$$

$$y = \left(\phi + c_3\phi^3 + c_5\phi^5\right) \tag{4.33}$$

4.1.17 Urmayev Cylindrical III.

`+proj=urm_3C` Fig. 4.3

$$a_0 = 0.92813433 \qquad a_2 = 1.11426959 \tag{4.34}$$

$$x = R\lambda \qquad y = R\left(a_0\phi + \frac{a_2}{3}\phi^3\right) \tag{4.35}$$

4.2 Transverse and Oblique Aspects.

4.2.1 Transverse Mercator

`+proj=tmerc`
`+proj=utm`

Spherical

Forward projections:

$$B = \cos\phi \sin\lambda \tag{4.36}$$

$$x = \begin{cases} \frac{Rk_0}{2} \ln\left(\frac{1+B}{1-B}\right) \\ Rk_0 \tanh^{-1} B \end{cases} \quad y = Rk_0 \left[\arctan\left(\frac{\tan\phi}{\cos\lambda}\right) - \phi_0\right] \tag{4.37}$$

Inverse projection:

$$D = \frac{y}{Rk_0} + \phi_0 \tag{4.38}$$

$$\phi = \arcsin\left(\frac{\sin D}{\cosh x'}\right) \tag{4.39}$$

$$\lambda = \arctan\left(\frac{\sinh x'}{cosD}\right) \tag{4.40}$$

$$x' = \frac{x}{Rk_0} \tag{4.41}$$

Transverse Mercator – Gauss-Krüger

Forward projection:

$$\begin{aligned} \frac{x}{N} = \lambda\cos\phi &+ \frac{\lambda^3\cos^3\phi}{3!}(1 - t^2 + \eta^2) \\ &+ \frac{\lambda^5\cos^5\phi}{5!}(5 - 18t^2 + t^4 + 14\eta^2 - 58t^2\eta^2) \\ &+ \frac{\lambda^7\cos^7\phi}{7!}(61 - 479t^2 + 179t^4 - t^6) \end{aligned} \tag{4.42}$$

$$\begin{aligned} \frac{y}{N} = \frac{M(\phi)}{N} &+ \frac{\lambda^2\sin\phi\cos\phi}{2!} \\ &+ \frac{\lambda^4\sin\phi\cos^3\phi}{4!}(5 - t^2 + 9\eta^2 + 4\eta^4) \\ &+ \frac{\lambda^6\sin\phi\cos^5\phi}{6!}(61 - 58t^2 + t^4 + 270\eta^2 - 330t^2\eta^2) \\ &+ \frac{\lambda^8\sin\phi\cos^7\phi}{8!}(1,385 - 3,111t^2 + 543t^4 - t^6) \end{aligned} \tag{4.43}$$

where

$$N = \frac{a}{(1 - e^2\sin^2\phi)^{1/2}} \tag{4.44}$$

$$R = \frac{a(1 - e^2)}{(1 - e^2\sin^2\phi)^{3/2}} \tag{4.45}$$

$$t = \tan\phi \tag{4.46}$$

$$\eta^2 = \frac{e^2}{1 - e^2}\cos^2\phi \tag{4.47}$$

and where $M(\phi)$ is the meridianal distance. Inverse projection: Given the "footprint" latitude $\phi_1 = M^{-1}(y)$:

$$
\begin{aligned}
\phi = \phi_1 &- \frac{t_1 x^2}{2! R_1 N_1} + \frac{t_1 x^4}{4! R_1 N_1^3}(5 + 3t_1^2 + \eta_1^2 - 4\eta_1^4 - 9\eta_1^2 t_1^2) \\
&- \frac{t_1 x^6}{6! R_1 N_1^5}\left(\begin{array}{c} 61 + 90t_1^2 + 46\eta_1^2 + 45t_1^4 - 252t_1^2\eta_1^2 \\ -3\eta_1^4 + 100\eta_1^6 - 66t_1^2\eta_1^4 - 90t_1^4\eta_1^2 i \\ +88\eta_1^8 + 225t_1^4\eta_1^4 + 84t_1^2\eta_1^6 - 192t_1^2\eta_1^8 \end{array}\right) \\
&+ \frac{t_1 x^8}{8! R_1 N_1^7}(1,385 + 3,633t_1^2 + 4,095t_1^4 + 1,575t_1^6)
\end{aligned}
\tag{4.48}
$$

$$
\begin{aligned}
\lambda = &\frac{x}{\cos\phi N_1} - \frac{x^3}{3! \cos\phi N_1^3}(1 + 2t_1^2 + \eta_1^2) \\
&+ \frac{x^5}{5! \cos\phi N_1^5}\left(\begin{array}{c} 5 + 6\eta_1^2 + 28t_1^2 - 3\eta_1^4 + 8t_1^2\eta_1^2 \\ +24t_1^4 - 4\eta_1^6 + 4t_1^2\eta_1^4 + 24t_1^2\eta_1^6 \end{array}\right) \\
&- \frac{x^7}{7! \cos\phi N_1^7}(61 + 662t_1^2 + 1,320t_1^4 + 720t_1^6)
\end{aligned}
\tag{4.49}
$$

4.2.2 Gauss-Boaga

Forward projection:

$$
\begin{aligned}
\frac{x}{N} &= \lambda\cos\phi + \frac{\lambda^3\cos^3\phi}{3!}(1 - t^2 + \eta^2) \\
&\quad + \frac{\lambda^5\cos^5\phi}{5!}(5 - 18t^2 + t^4 + 14\eta^2 - 58t^2\eta^2)
\end{aligned}
\tag{4.50}
$$

$$
\begin{aligned}
\frac{y}{N} &= \frac{M(\phi)}{N} + \frac{\lambda^2\sin\phi\cos\phi}{2!} \\
&\quad + \frac{\lambda^4\sin\phi\cos^3\phi}{4!}(5 - t^2 + 9\eta^2 + 4\eta^4) \\
&\quad + \frac{\lambda^6\sin\phi\cos^5\phi}{6!}(61 - 58t^2 + t^4 + 270\eta^2 - 330t^2\eta^2)
\end{aligned}
$$

Inverse projection:

$$
\begin{aligned}
\phi = \phi_1 &- \frac{x^2 t_1}{2! N_1^2}(1 + \eta_1^2) \\
&+ \frac{x^4 t_1}{4! N_1^4}(5 + 3t_1^2 + 6\eta_1^2 - 6t_1^2\eta_1^2 - 3\eta_1^4 - 9t_1^2\eta_1^4 \\
&- \frac{x^6 t_1}{6! N_1^6}(61 + 90t_1^2 + 45t_1^4 + 107\eta_1^2 - 162t_1^2\eta_1^2 - 45t_1^4\eta_1^2)
\end{aligned}
\tag{4.51}
$$

$$
\begin{aligned}
\lambda = &\frac{x}{\cos\phi_1 N_1} - \frac{x^3}{3! \cos\phi_1 N_1^3}(1 + 2t_1^2 + \eta_1^2) \\
&+ \frac{x^5}{5! \cos\phi_1 N_1^5}(5 + 28t_1^2 + 24t_1^4 + 6\eta_1^2 + 8t_1^2\eta_1^2)
\end{aligned}
\tag{4.52}
$$

4.2.3 Oblique Mercator

`+proj=omerc` (see below for full list of options)
The oblique Mercator projection is designed for elongated regions aligned along a geodesic[1] arc (Great Circle) where the cylinder of the projection is tangent to the

[1] The centerline is a true geodesic only for the spherical case and approximates a geodesic in the ellipsoidal case.

sphere or ellipsoid ($k_0 = 1$). Ellipsoid equations presented here are based upon Snyder's [14, p. 66–75] development of Hotine's [7] "rectified skewed orthomorphic" projection and a development found in material by EPSGr [2]. In none of these sources were the developments sufficiently complete to perform projections of several common grid systems and it was necessary to merge operations to create a more general procedure.

Two methods are used to specify the projection parameters: by specifying two points that lay on the centerline of the projection or by specifying the geographic coordinates of the central point on the centerline and specifying an azimuth of the centerline. The latter method is most commonly used for grid systems.

Two point method

Parameters of the two-point methods are as follows:

`lat_1=`	(ϕ_1, λ_1) latitude and longitude of the
`lon_1=`	first point on the centerline
`lat_2=`	(ϕ_2, λ_2) latitude and longitude of the
`lon_2=`	second point on the centerline
`lat_0=`	ϕ_0 latitude of the center of the map
`k_0=`	k_0 scale factor along the centerline
`no_rot`	if present, do not rotate axis

Note that the central meridian (`lon_0`) common to most projections is not determined by the user. Restrictions on parameter specification is such that a centerline may not coincide with a meridian (Transverse Mercator case) nor coincide with the equator (simple Mercator case). Also, $\phi_1 \neq \phi_2$. First, compute factors common to both control specification method. For $\phi_0 \neq 0$ then

$$B = \left(1 + \frac{e^2}{1-e^2}\cos^4\phi_0\right)^{\frac{1}{2}} \tag{4.53}$$

$$A = Bk_0\frac{(1-e^2)^{\frac{1}{2}}}{1-e^2\sin^2\phi_0} \tag{4.54}$$

$$t_0 = \Psi(\phi_0) \tag{4.55}$$

$$D = \frac{B(1-e^2)^{\frac{1}{2}}}{\cos\phi_0(1-e^2\sin^2\phi_0)^{\frac{1}{2}}} \tag{4.56}$$

$$F = D \pm \sqrt{D^2-1} \quad \text{taking sign of } \phi_0 \tag{4.57}$$

$$E = t_0^B F \tag{4.58}$$

where $\Psi()$ is the Isometric Latitude kernel function (`pj_tsfn`). Set $D = 1$ if $D^2 < 1$. other wise

$$B = (1-e^2)^{-\frac{1}{2}} \qquad A = k_0 \qquad E = D = F = 1 \tag{4.59}$$

Now continue with initialization unique to the two point method:

$$t_1 = \Psi(\phi_1) \tag{4.60}$$

$$t_2 = \Psi(\phi_2) \tag{4.61}$$

$$H = t_1^B \tag{4.62}$$

$$L = t_2^B \tag{4.63}$$

$$F = \frac{E}{H} \tag{4.64}$$

$$G = (F - 1/F)/2 \tag{4.65}$$

$$J = \frac{E^2 - LH}{E^2 + LH} \tag{4.66}$$

$$P = \frac{L - H}{L + H} \tag{4.67}$$

$$\lambda_0 = \frac{\lambda_1 + \lambda_2}{2} - \frac{1}{B} \arctan\left(\frac{J}{P} \tan\left[\frac{B}{2}(\lambda_1 - \lambda_2)\right]\right) \tag{4.68}$$

$$\gamma_0 = \arctan\left(\frac{\sin(B(\lambda_1 - \lambda_0))}{G}\right) \tag{4.69}$$

$$\alpha_c = \arcsin(D \sin\gamma_0) \tag{4.70}$$

Unless `no_rot` is specified the axis rotation γ is set from α_c and rotation is about the ϕ_0 position.

Central point and azimuth method

The parameters for this case are:

`lat_0=` `lonc=`	(ϕ_0, λ_c) latitude and longitude of the central point of the line.
`alpha=`	α_c azimuth of centerline clockwise from north at the center point of the line. If `gamma` is not given then α_c determines the value of γ.
`gamma=`	γ azimuth of centerline clockwise from north of the rectified bearing of centre line. If `alpha` is not given, then `gamma` is assign to γ_0 from which α_c is derived (see equation 4.71).
`k_0=`	k_0 scale factor along the centerline
`ro_rot`	if present, do not rotate axis
`no_off`	if present, do not offset origin to center of projection ($u_0 = 0$).

. To determine initialization parameters for this specification form of the projection first determine B, A, t_0, D, F and E from equations 4.53 through 4.58 and then proceed as follows:

$$\sin\alpha_c = D \sin\gamma_0 \tag{4.71}$$

$$G = \frac{F - 1/F}{2} \tag{4.72}$$

$$\lambda_0 = \lambda_c - \frac{\arcsin(G \tan\gamma_0)}{B} \tag{4.73}$$

Common Initialization

If `no_off` is specified then

$$u_c = 0$$

otherwise the u axis is corrected by:

$$u_c = \pm\frac{A}{B}\text{atan2}(\sqrt{D^2-1}, \cos\alpha_c) \tag{4.74}$$

taking the sign of ϕ_0.

Forward elliptical projection

The first phase is to convert the geographic coordinates (ϕ, λ) to the intermediate Cartesian system (u, v) where the u axis is coincident with the centerline of the projection and the projection (u, v) system origin is at the aposphere equator and longitude λ_0.

First compute

$$V = \sin[B(\lambda - \lambda_0)] \tag{4.75}$$

If $|\phi| \neq \pi/2$ then:

$$Q = \frac{E}{\Psi(\phi)^B} \tag{4.76}$$

$$S = \frac{Q - 1/Q}{2} \tag{4.77}$$

$$T = \frac{Q + 1/Q}{2} \tag{4.78}$$

$$U = \frac{-V\cos\gamma_0 + S\sin\gamma_0}{T} \tag{4.79}$$

$$v = \begin{cases} \dfrac{A}{2B}\ln\left(\dfrac{1-U}{1+U}\right) & |U| \neq 1 \\ \infty & |U| = 1 \end{cases} \tag{4.80}$$

$$M = \cos[B(\lambda - \lambda_0)] \tag{4.81}$$

$$u = \begin{cases} \dfrac{A}{B}\text{atan2}(S\cos\gamma_0 + V\sin\gamma_0, M) & M \neq 0 \\ AB(\lambda - \lambda_0) & M = 0 \end{cases} \tag{4.82}$$

otherwise:

$$v = \frac{A}{B}\ln\tan\left(\frac{\pi}{4} \mp \frac{\gamma_0}{2}\right) \qquad u = \phi\frac{A}{B} \tag{4.83}$$

If rotation is suppressed by the `no_rot` option then

$$x = u \qquad y = v \tag{4.84}$$

else

$$u = -u_c \qquad x = v\cos\gamma + u\sin\gamma \qquad y = u\cos\gamma - v\sin\gamma \tag{4.85}$$

Inverse elliptical projection

First rotate (x, y) system into (u, v) system:

$$v = x\cos\gamma - y\sin\gamma \qquad u = y\cos\gamma + x\sin\gamma + u_c \tag{4.86}$$

$$Q' = \exp\left(-\frac{Bv}{A}\right) \tag{4.87}$$

$$S' = \frac{Q' - 1/Q'}{2} \tag{4.88}$$

$$T' = \frac{Q' + 1/Q'}{2} \tag{4.89}$$

$$V' = \sin\left(\frac{Bu}{A}\right) \tag{4.90}$$

$$U' = \frac{V'\cos\gamma_0 + S'\sin\gamma_0}{T'} \tag{4.91}$$

If $|U'| = 1$, then $\phi = \pm\pi/2$ taking sign of U' and $\lambda = \lambda_0$. Otherwise

$$t = \left[E\left(\frac{1-U'}{1+U'}\right)\frac{1}{2}\right]^{\frac{1}{B}} \tag{4.92}$$

$$\phi = \frac{\pi}{2} - 2\arctan\left[t\left(\frac{1-e\sin\phi}{1+e\sin\phi}\right)^{\frac{e}{2}}\right] \tag{4.93}$$

$$\lambda = \frac{1}{B}\text{atan2}\left[S'\cos\gamma_0 - V'\sin\gamma_0, \cos\left(\frac{Bu}{A}\right)\right] \tag{4.94}$$

where equation 4.93 is solved by iteration in function `pj_phi2`.

Examples.

The first example of this projection is the Timbalai 1948/R̈.S.O. Borneo grid system from EPSG [2][p. 35–36] defined by:

```
proj=omerc a=6377298.556 rf=300.8017
lat_0=4 lonc=115 alpha=53d18'56.9537
gamma=53d7'48.3685 k_0=0.99984
x_0=590476.87 y_0=442857.65
```

Lon/lat	Easting/Northing
115d48'19.8196"E	679245.73
5d23'14.1129"N	596562.78

Zone 1 of the Alaska State Plane Coordinate System uses the Oblique Mercator projection as in this NAD27 example:

```
proj=omerc  a=6378206.4
es=.006768657997291094
k=.9999 lonc=-133d40 lat_0=57
alpha=-36d52'11.6315
x_0=818585.5672270928 y_0=575219.2451072642
units=us-ft
```

Lon/lat	Easting/Northing
-134d00'00.000"	2615716.535
55d00'00.000"	1156768.938

The values agree with those computed by GCTP [21, 20].

4.2.4 Cassini.

`+proj=cass` Ref. [14, p. 94–95]

Spherical form.

Forward projection:

$$x = \arcsin(\cos\phi \sin\lambda) \qquad y = \operatorname{atan2}(\tan\phi, \cos\lambda) - \phi_0 \tag{4.95}$$

Inverse projection:

$$\phi = \arcsin\left[\sin(y + \phi_0)\cos x\right] \qquad \lambda = \operatorname{atan2}\left(\tan x, \cos(y + \phi_0)\right) \tag{4.96}$$

Elliptical form.

Forward projection:

$$N = (1 - e^2 \sin^2\phi)^{-1/2} \tag{4.97}$$
$$T = \tan^2\phi \tag{4.98}$$
$$A = \lambda\cos\phi \tag{4.99}$$
$$C = \frac{e^2}{1 - e^2}\cos^2\phi \tag{4.100}$$
$$x = N\left(A - T\frac{A^3}{6} - (8 - T + 8C)T\frac{A^5}{120}\right) \tag{4.101}$$
$$y = \mathrm{M}(\phi) - \mathrm{M}(\phi_0) + N\tan\phi\left(\frac{A^2}{2} + (5 - T + 6C)\frac{A^4}{24}\right) \tag{4.102}$$

where M() us the meridianal distance function (3.2). Inverse projection:

$$\phi' = \mathrm{M}^{-1}(\mathrm{M}(\phi_0) + y) \tag{4.103}$$

If $\phi' = \pi/2$ then $\phi = \phi'$ and $\lambda = 0$ otherwise evaluate T and N above using ϕ' and

$$R = (1 - e^2)(1 - e^2\sin^2\phi')^{-3/2} \tag{4.104}$$
$$D = x/N \tag{4.105}$$
$$\phi = \phi' - \tan\phi'\frac{N}{R}\left(\frac{D^2}{2} - (1 + 3T)\frac{D^4}{24}\right) \tag{4.106}$$
$$\lambda = \left(D - T\frac{D^3}{3} + (1 + 3T)T\frac{D^5}{15}\right)/\cos\phi' \tag{4.107}$$

4.2.5 Swiss Oblique Mercator Projection

`+proj=somerc` [1]

The Swiss Oblique Mercator Projection (a tentative name based upon the Swiss usage in their CH1903 grid system) is based upon a three step process:

1. conformal transformation of ellipsoid coordinates to a sphere,
2. rotational translation of the spherical system so that the specified projection origin will lie on the equator, and
3. Mercator projection of geographic coordinates to the Cartesian system.

The projection cylinder is tangent at the projection origin (λ_0, ϕ_0) with zero scale error at the projection origin $(k_0 = 1)$ with minimum error extending east-west near the central meridian. In this projection, axis rotation only occurs about an axis normal to the plane of the central meridian (Wray's "simple oblique aspect" [8, pages 135–138]).

For the forward projection the input geographic coordinates are processed in following manner:

$$(\lambda, \phi) \rightarrow \texttt{pj\_gauss} \rightarrow \texttt{pj\_translate} \rightarrow (\lambda', \phi')$$

where `pj_gauss` (3.3) and `pj_translate` (3.5) are the respective conversion to Gaussian sphere and axis translation-rotation procedures. Then standard, spherical Mercator projection (4.2) is applied in-line for conversion to (x, y). Final scaling is performed by multiplying the radius of the conformal sphere, returned by the Gauss initialization, and with k_0.

Inverse projection follows the reverse sequence of the above steps by using the inverse Mercator projection, inverse of spherical coordinate transformation and inverse from the Gaussian sphere to the ellipsoid coordinates.

The following example demonstrates the example from [1, p. 9] where the control parameters are

```
+proj=somerc
+ellps=bessel
+lon_0=7d26'22.50
+lat_0=46d57'08.66
+x_0=2600000
+y_0=1200000
```

and geographic and Swiss projection coordinates are:

$$\lambda = 8°9'11.11127154''E \leftrightarrow 2679520.05 \text{ Easting} \tag{4.108}$$
$$\phi = 47°03'28.95659233''N \leftrightarrow 1212273.44 \text{ Northing} \tag{4.109}$$

This projection has general application for grid system that have proportionally longer extensions along the Easting.

4.2.6 Laborde.

`+proj=labrd +azi=`

The Laborde projection was developed and exclusively used for the Madagascar Grid System with these parameters:

```
+proj=labrd
+azi=18d54'
+lat_0=18d54'S
+lon_0=46d26'13.95"E
+k_0=0.9995
+x_0=400000
+y_0=800000
+ellps=intnl
```

This projection should not be confused with the Hotine Oblique Mercator nor should the later be used as a substitute. [15, p. 162].

The following are initialization steps:

$$
\begin{aligned}
R &= (1-e^2)(1-e^2\sin^2\phi)^{-3/2} && (4.110)\\
N &= (1-e^2\sin^2\phi)^{-1/2} && (4.111)\\
R_g &= (NR)^{1/2}\text{geometric mean for radius of Gauss sphere} && (4.112)\\
\phi_{0s} &= \arctan((R_0/N_0)^{1/2}\tan\phi_0) && (4.113)\\
A &= \sin\phi_0/\sin\phi_{0s} && (4.114)\\
C &= \frac{eA}{2}\ln\frac{1+e\sin\phi_0}{1-e\sin\phi_0} - A\ln\tan(\pi/4+\phi_0/2) + \ln\tan(\pi/4+\phi_{0s}/2) && (4.115)\\
C_a &= \frac{1-\cos 2A_z}{12R_g^2k_0^2} && (4.116)\\
C_b &= \frac{\sin 2A_z}{12R_g^2k_0^2} && (4.117)\\
C_c &= 3(C_a^2 - C_b^2) && (4.118)\\
C_d &= 6C_aC_b && (4.119)
\end{aligned}
$$

Forward computations:

$$
\begin{aligned}
V_1 &= A\ln\tan(\pi/4+\phi/2) && (4.120)\\
V_2 &= \frac{eA}{2}\ln\frac{1+e\sin\phi}{1-e\sin\phi} && (4.121)\\
\phi_s &= 2(\tan^{-1}\exp(V_1-V_2+C)-\pi/4) && (4.122)\\
I_1 &= \phi_s - \phi_{0s} && (4.123)\\
I_2 &= A^2\sin\phi_s\cos\phi_s/2 && (4.124)\\
I_3 &= A^4\sin\phi_s\cos^3\phi_s(5-\tan^2\phi_s)/24 && (4.125)\\
&= A^4\sin\phi_s\cos\phi_s(5\cos^2\phi_s-\sin^2\phi_s)/24 && (4.126)\\
I_4 &= A\cos\phi_s && (4.127)\\
I_5 &= A^3\cos^3\phi_s(1-\tan^2\phi_s)/6 && (4.128)\\
&= A^3\cos\phi_s(\cos^2\phi_s-\sin^2\phi_s)/6 && (4.129)\\
I_6 &= A^5\cos^5\phi_s(5-18\tan^2\phi_s+\tan^4\phi_s)/120 && (4.130)\\
&= A^5\cos\phi_s(5\cos^4\phi_s-18\cos^2\phi_s\sin^2\phi_s+\sin^4\phi_s)/120 && (4.131)\\
x_g &= k_0R_g\lambda(I_4+\lambda^2(I_5+\lambda^2I_6)) && (4.132)\\
y_g &= k_0R_g(I_1+\lambda^2(I_2+\lambda^2I_3)) && (4.133)\\
V_1 &= 3x_gy_g^2 - x_g^3 && (4.134)\\
V_2 &= y_g^3 - 3x_g^2y_g && (4.135)\\
x &= x_g + C_aV_1 + C_bV_2 && (4.136)\\
y &= y_g - C_bV_1 + C_aV_2 && (4.137)
\end{aligned}
$$

Inverse formulas:

$$V_1 = 3xy^2 - x^3 \tag{4.138}$$
$$V_2 = y^3 - 3x^2y \tag{4.139}$$
$$V_3 = x^5 - 10x^3y^2 + 5xy^4 \tag{4.140}$$
$$V_4 = 5x^4y - 10x^2y^3 + y^5 \tag{4.141}$$
$$x_g = x - C_aV_1 - C_bV_2 + C_cV_3 + C_dV_4 \tag{4.142}$$
$$y_g = y + C_bV_1 - C_aV_2 - C_dV_3 + C_cV_4 \tag{4.143}$$
$$\phi_s = \phi_{0s} + y_g/(R_gk_0) \tag{4.144}$$
$$\phi_e = \phi_s + \phi_0 - \phi_{0s} \tag{4.145}$$

Iterate

$$\tag{4.146}$$
$$V_1 = A \ln \tan(\pi/4 + \phi_e/2) \tag{4.147}$$
$$V_2 = \frac{eA}{2} \ln \frac{1 + e\sin\phi_e}{1 - e\sin\phi_e} \tag{4.148}$$
$$t = \phi_s - 2(\tan^{-1}\exp(V_1 - V_2 + C) - \pi/4) \tag{4.149}$$
$$\phi_e = \phi_e + t \tag{4.150}$$

until $|t| < \epsilon$

$$\tag{4.151}$$
$$R_e = a(1-e^2)(1-e^2\sin^2\phi_e)^{-3/2} \tag{4.152}$$
$$I_7 = \tan\phi_s/(2R_eR_gk_0^2) \tag{4.153}$$
$$I_8 = \tan\phi_s(5 + 3\tan^2\phi_s)/(24R_eR_g^3k_0^4) \tag{4.154}$$
$$I_9 = 1/(\cos\phi_sR_gk_0A) \tag{4.155}$$
$$I_{10} = (1 + 2\tan^2\phi_s)/(6\cos\phi_sR_g^3k_0^3A) \tag{4.156}$$
$$I_{11} = (5 + 28\tan^2\phi_s + 24\tan^4\phi_s)/(120\cos\phi_sR_g^5k_0^5A) \tag{4.157}$$
$$\phi = \phi_e - I_7x_g^2 + I_8x_g^4 \tag{4.158}$$
$$\lambda = I_9x_g - I_{10}x_g^3 + I_{11}x_g^5 \tag{4.159}$$

Chapter 5

Pseudocylindrical Projections

Pseudocylindrical projections have the mathematical characteristics of

$$\begin{aligned} x &= f(\lambda, \phi) \\ y &= g(\phi) \end{aligned}$$

where the parallels of latitude are straight lines, like cylindrical projections, but the meridians are curved toward the center as they depart from the equator. This is an effort to minimize the distortion of the polar regions inherent in the cylindrical projections. Pseudocylindrical projections are almost exclusively used for small scale global displays and, except for the Sinusoidal projection, only derived for a spherical Earth. Because of the basic definition none of the pseudocylindrical projections are conformal but many are equal area.

To further reduce distortion, pseudocylindrical are often presented in interrupted form that are made by joining several regions with appropriate central meridians and false easting and clipping boundaries. Figure 5.1 shows typical constructions that are suited for showing respective global land and oceanic regions. To reduce the lateral size of the map, some uses remove an irregular, North-South strip of the mid-Atlantic region so that the western tip of Africa is plotted north of the eastern tip of South America.

5.1 Computations.

A complicating factor in computing the forward projection for pseudocylindricals is that some of the projection formulas use a parametric variable, typically θ, which is a function of ϕ. In some cases, the parametric equation is not directly solvable for θ and requires use of Newton-Raphson's method of iterative finding the root of $P(\theta)$. The defining equations for these cases are thus given in the form of $P(\theta)$ and its derivative, $P'(\theta)$, and an estimating initial value for $\theta_0 = f(\phi)$. Refinement of θ is made by $\theta \leftarrow \theta - P(\theta)/P'(\theta)$ until $|P(\theta)/P'(\theta)|$ is less than predefined tolerance.

When known, formula constant factors are given in rational form (e.g. $\sqrt{2}/2$) rather than a decimal value (0.7071) so that the precision used in the resultant program code constants is determined by the programmer. However, source material may only provide decimal values, typically to 5 or 6 decimal digits. This is adequate in most cases, but has caused problems with the convergence of a Newton-Raphson determination and degrades the determination of numerical derivatives.

Because several of the pseudocylindrical projections have a common computational base, they are grouped into a single module with multiple initializing entry

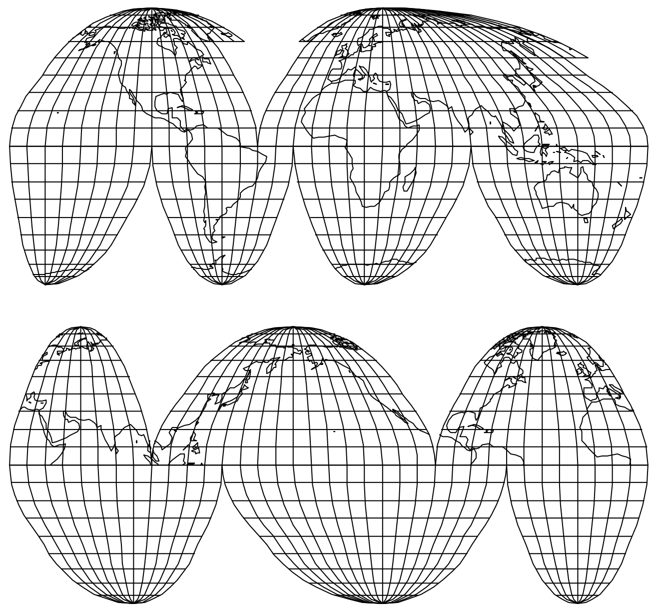

Figure 5.1: Interupted Projections.
Interupted Goode Homolosine: A–continental regions, B–oceanic regions.

points. This may lead to a minor loss of efficiency, such as adding a zero term in the simple Sinusoidal case of the the Generalized Sinusoidal.

5.2 Spherical Forms.

5.2.1 Sinusoidal.

Equal-area for all cases.

Name	+proj=	figure	Ref.
General Sinusoidal	`gn_sinu +m= +n=`		
Sinusoidal Sanson-Flamsteed	`sinu`	5.2	[14, p. 243-248]
Eckert VI	`eck4`	5.3	[17, p. 220]
McBryde-Thomas Flat-Polar Sinusoidal	`mbtfps`	5.7	[17, p. 220]

$$x = C\lambda(m + \cos\theta)/(m + 1) \tag{5.1}$$
$$y = C\theta \tag{5.2}$$
$$C = \sqrt{(m + 1)/n} \tag{5.3}$$
$$P(\theta) = m\theta + \sin\theta - n\sin\phi \tag{5.4}$$
$$P'(\theta) = m + cos\theta \tag{5.5}$$
$$\theta_0 = \phi \tag{5.6}$$

	m	n	C
Sinusoidal (Sanson-Flamsteed)	0	1	1
Eckert VI	1	$1 + \pi/2$	$2/\sqrt{2 + \pi}$
McBryde-Thomas Flat-Polar Sinusoidal	1/2	$1 + \pi/4$	$\sqrt{6/(4 + \pi)}$

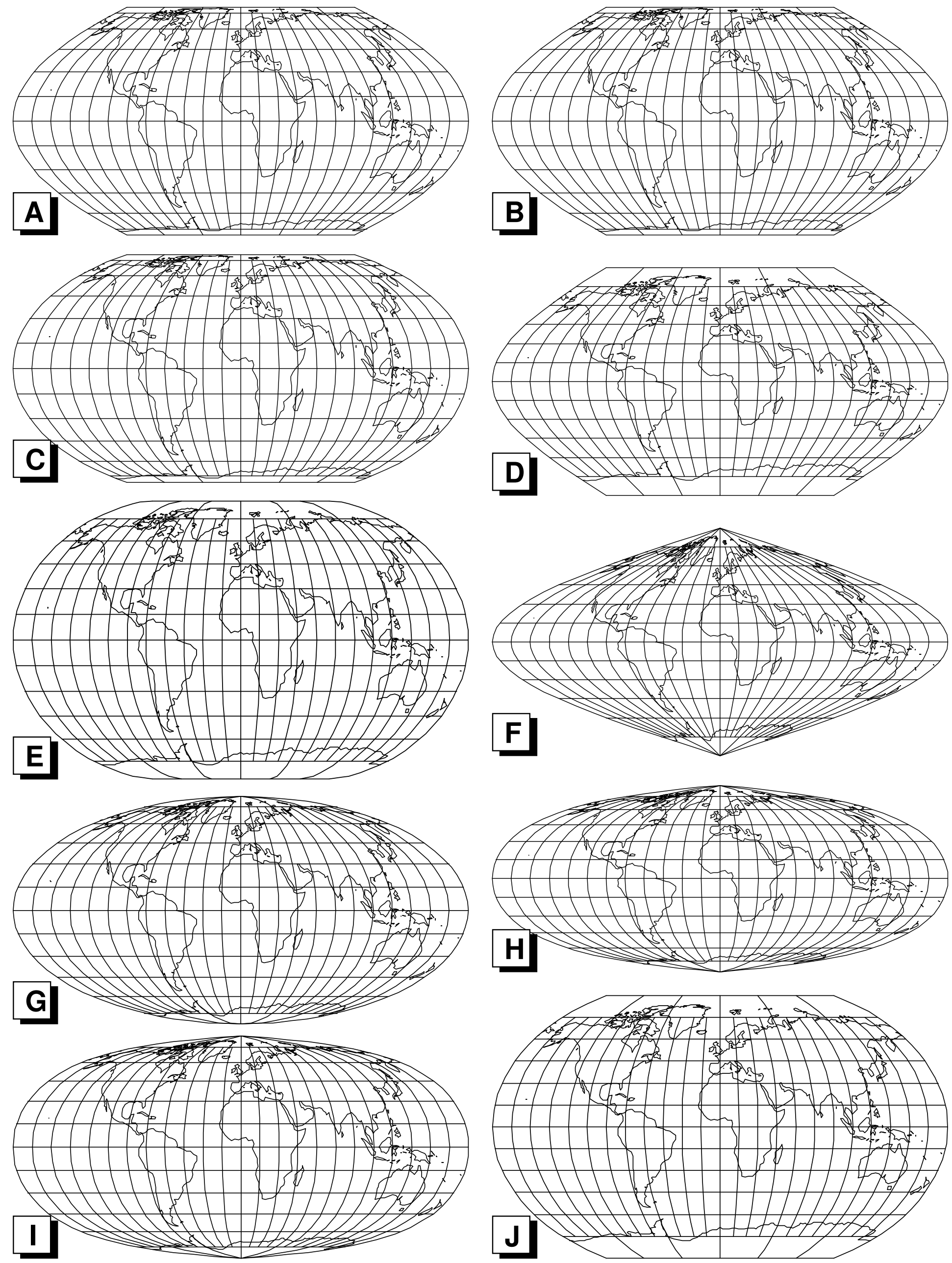

Figure 5.2: General pseudocylindricals I
A–Werenskiold I, **B**–Werenskiold II, **C**–Werenskiold III, **D**–Winkel I, **E**–Winkel II (+lat_1=50d28'), **F**–Sinusoidal, **G**–Mollweide, **H**–Foucaut Sinusoidal (+n=0.5), **I**–Kavraisky V and **J**–Kavraisky VII .

5.2.2 Winkel I.

+proj=wink1 +lat_ts= Fig. 5.2 Ref. [17, p. 220]
Option lat_ts=ϕ_{ts} estabishes latitude of true scale on central meridian (default $= 0^\circ$ and thus the same as Eckert V). Not equal-area but if $\cos\phi_{ts} = 2/\pi$ (lat_ts=50d28) the total area of the global map is correct.

$$x = \lambda(\cos\phi_{ts} + \cos\phi)/2 \qquad y = \phi \tag{5.7}$$

5.2.3 Winkel II.

`+proj=wink2 +lat_1=` Fig. 5.2 Ref. [13, p. 77]
Arithmetic mean of Equirectangular and Mollweide and is not equal-area. Parameter `lat_1=`ϕ_1 controls standard parallel and width of flat polar extent.

$$x = \lambda(\cos\theta + \cos\phi_1)/2 \qquad y = \pi(\sin\theta + 2\phi/\pi)/4 \tag{5.8}$$
$$P(\theta) = 2\theta + \sin 2\theta - \pi\sin\phi \qquad P'(\theta) = 2 + 2\cos 2\theta \tag{5.9}$$
$$\theta_0 = 0.9\phi \tag{5.10}$$

As with Mollweide, P converges slowly as $\phi \to \pi/2$ and $\theta \to \pi/2$.

5.2.4 Urmayev Flat-Polar Sinusoidal Series.

Urmaev and Wagner are equal area but Werenskiold has true scale at the equator.

Name	`+proj=`	Fig.	Ref.
Urmayev FPS	urmfps +n=	5.8	[13][p. 62]
Wagner I (Kavraisky VI)	wag1	5.4	[22]
Werenskiold II	weren2	5.2	[13][p. 62]

	C_x	C_y	C_n
Urmayev FPS	$\frac{2\sqrt[4]{3}}{3}$	$\frac{1}{nC_x}$	$0 \le n \le 1$
Wagner I (Kavraisky VI)	$\frac{2\sqrt[4]{3}}{3}$	$\sqrt[4]{3}$	$\frac{\sqrt{3}}{2}$
Werenskiold II	$\frac{3^{0.75}}{2}\cdot\frac{2\sqrt[4]{3}}{3}$	$\frac{3^{0.75}}{2}\cdot\sqrt[4]{3}$	$\frac{\sqrt{3}}{2}$

$$x = C_x\lambda\cos\psi \qquad y = C_y\psi \tag{5.11}$$
$$\sin\psi = C_n\sin\phi \tag{5.12}$$

For Urmayev the latitudes of true scale are determined by the relation:

$$\phi_{ts} = \arcsin\sqrt{\frac{9-4\sqrt{3}}{9-4n^2\sqrt{3}}} \tag{5.13}$$

and the ratio of the length of the poles to the equator is determined by $\sqrt{1-n^2}$.

5.2.5 Eckert I.

`+proj=eck1` Fig. 5.3 Ref. [**?** , p. 223]

$$x = 2\sqrt{2/3\pi}\lambda(1 - |\phi|/\pi) \qquad y = 2\sqrt{2/3\pi}\phi \tag{5.14}$$

5.2.6 Eckert II.

`+proj=eck2` Fig. 5.3 Ref. [17, p. 223]

$$x = (2/\sqrt{6\pi})\lambda\sqrt{4-3\sin|\phi|} \qquad y = \pm\left(\sqrt{2\pi/3}(2 - \sqrt{4-3\sin|\phi|})\right) \tag{5.15}$$

where y assumes sign of ϕ.

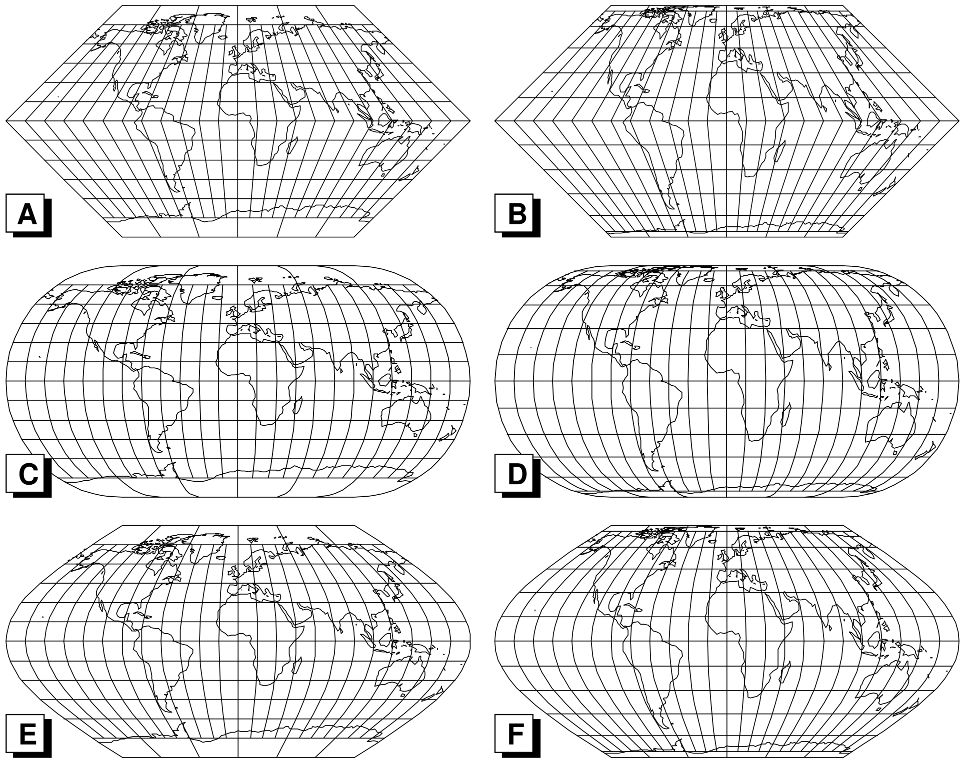

Figure 5.3: Eckert pseudocylindrical series
A–Eckert I, **B**–II, **C**–III, **D**–IV, **E**–V and **F**–VI.

5.2.7 Eckert III, Putniņš P_1, Putniņš P_1', Wagner VI and Kavraisky VII.

None of these projections are equal-area and are flat-polar when coefficient $A \neq 0$.

Name	`+proj=`	figure	Ref.
Eckert III	`eck3`	5.3	
Putniņš P_1	`putp1`	5.6	
Putniņš P_1'	`putp1p`	5.6	
Wagner VI	`wag6`	5.4	
Kavraisky VII	`kav7`	5.2	[13][p. 67]

$$x = C_x \lambda (A + \sqrt{1 - B(\phi/\pi)^2}) \qquad y = C_y \phi \tag{5.16}$$

	C_x	C_y	A	B
Putniņš P_1	1.89490	0.94745	$-1/2$	3
Putniņš P_1'	$\frac{1.89490}{2}$	0.94745	0	3
Wagner VI	1	1	0	3
Eckert III	$\frac{2}{\sqrt{\pi(4+\pi)}}$	$\frac{4}{\sqrt{\pi(4+\pi)}}$	1	4
Kavraisky VII	$\sqrt{3}/2$	1	0	3

5.2.8 Eckert IV.

`+proj=eck4` Fig. 5.3 Ref. [17, p. 221]

$$x = 2\lambda(1+\cos\theta)/\sqrt{\pi(4+\pi)} \tag{5.17}$$
$$y = 2\sqrt{\pi/(4+\pi)}\sin\theta \tag{5.18}$$
$$\begin{aligned} P(\theta) &= \theta + \sin 2\theta + 2\sin\theta - \frac{(4+\pi)}{2}\sin\phi \qquad (5.19)\\ &= \theta + \sin\theta(\cos\theta + 2) - \frac{(4+\pi)}{2}\sin\phi \end{aligned}$$
$$\begin{aligned} P'(\theta) &= 2 + 4\cos 2\theta + 4\cos\theta \qquad (5.20)\\ &= 1. + \cos\theta(\cos\theta+2) - \sin^2\theta \end{aligned}$$
$$\theta_0 = 0.895168\phi + 0.0218849\phi^3 + 0.00826809\phi^5 \tag{5.21}$$

5.2.9 Eckert V.

`+proj=eck5` Fig. 5.3 Ref. [17, p. 220]

$$x = \lambda(1+\cos\phi)/\sqrt{2+\pi} \qquad y = 2\phi/\sqrt{2+\pi} \tag{5.22}$$

5.2.10 Wagner II.

`+proj=wag2` Fig. 5.4 Ref. [22, p. 184–187], [13, p. 64]

$$x = \frac{n}{\sqrt{nm_1m_2}}\lambda\cos\psi \qquad y = \frac{1}{\sqrt{nm_1m_2}}\psi \tag{5.23}$$
$$\sin\psi = m_1\sin(m_2\phi) \qquad n = \frac{2}{3} \tag{5.24}$$
$$m_2 = \frac{\arccos(1.2\cos 60^\circ)}{60^\circ} \qquad m_1 = \frac{\sqrt{3}}{2\sin\left(m_2\frac{\pi}{2}\right)} \tag{5.25}$$

5.2.11 Wagner III.

`+proj=wag3` Fig. 5.4 Ref: [22, p. 189–190]

$$x = \frac{\cos\phi_{ts}}{\cos(2\phi_{ts}/3)}\lambda\cos(2\phi/3) \qquad y = \phi \tag{5.26}$$

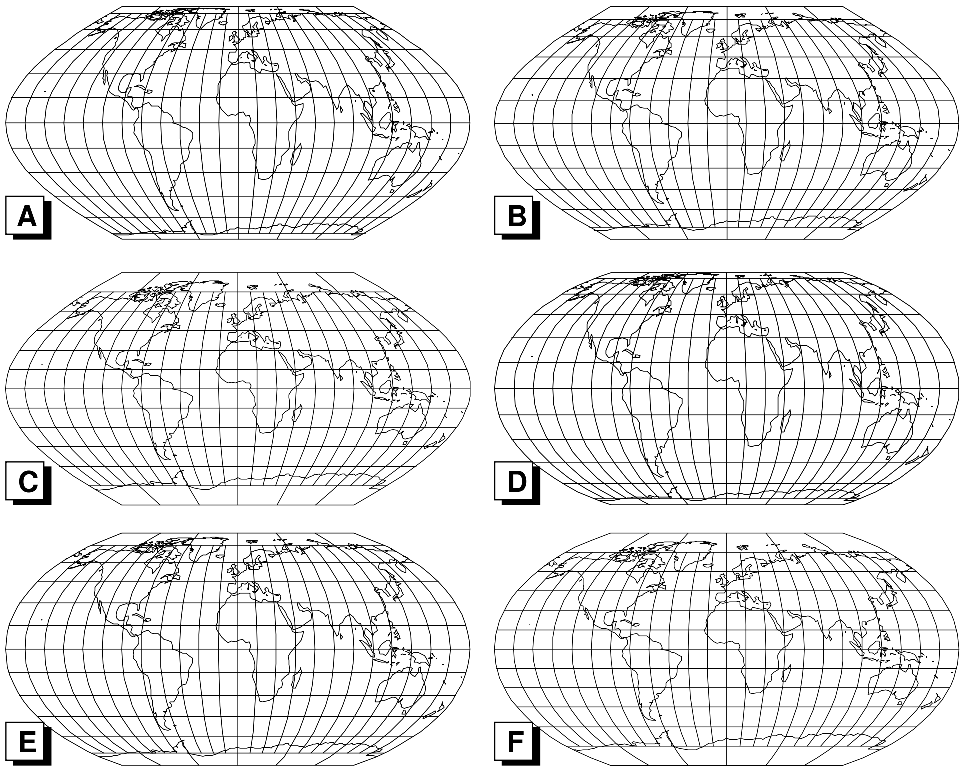

Figure 5.4: Wagner pseudocylindrical series
A–Wagner I, **B**–II, **C**–III, **D**–IV, **E**–V and **F**–VII

5.2.12 Wagner V.

`+proj=wag5` Fig. 5.4 Ref. [22, p. 194–196]

$$x = \frac{n2\sqrt{2}}{\pi\sqrt{nm_1n_1}}\lambda\cos\psi \tag{5.27}$$

$$y = \sqrt{\frac{2}{nm_1m_2}}\sin\psi \tag{5.28}$$

$$P(\psi) = 2\psi + \sin 2\psi - \pi m_1 \sin(m_2\phi) \tag{5.29}$$

$$P'(\psi) = 2 + 2\cos 2\psi \tag{5.30}$$

$$\psi_0 = \frac{2\phi}{3} \tag{5.31}$$

$$n = \frac{\sqrt{3}}{2} \tag{5.32}$$

$$m_2 = \frac{\arccos\left(1.2\cos 60^\circ\right)}{60^\circ} \tag{5.33}$$

$$m_1 = \frac{\frac{2\pi}{3} + \sin\frac{2\pi}{3}}{\pi\sin\left(\frac{m_2\pi}{2}\right)} \tag{5.34}$$

5.2.13 Foucaut Sinusoidal.

`+proj=fouc_s +n=` Fig. 5.2 Ref. [15][p. 113], [19]
An equal-area projection where the y-axis is a weighted arithmetic mean of the

Cylindrical Equal-Area and the Sinusoidal projections. Parameter `n=`n is the weighting factor where $0 \le n \le 1$.

$$x = \lambda \cos\phi / (n + (1-n)\cos\phi) \qquad y = n\phi + (1-n)\sin\phi \tag{5.35}$$

5.2.14 Mollweide, Bromley, Wagner IV (Putniņš P$'_2$) and Werenskiold III.

Mollweide and Wagner IV are equal area.

Name	+proj=	figure	Ref.
Mollweide	`moll`	5.2	[17, p. 220]
Bromley	`bromley`		[15, p. 163]
Wagner IV (Putniņš P$'_2$)	`wag4`	5.4	[22]
Werenskiold III	`weren3`	5.2	[13, p. 66]

$$x = C_x \lambda \cos(\theta) \tag{5.36}$$
$$y = C_y \sin(\theta) \tag{5.37}$$
$$P(\theta) = 2\theta + \sin 2\theta - C_p \sin\phi \tag{5.38}$$
$$P'(\theta) = 2 + 2\cos 2\theta \tag{5.39}$$
$$\theta_0 = \phi \tag{5.40}$$

	C_x	C_y	C_p
Mollweide	$\frac{2\sqrt{2}}{\pi}$	$\sqrt{2}$	π
Bromley	1	$\frac{4}{\pi}$	π
Wagner IV (Putniņš P$'_2$)	$\frac{2}{\pi}\sqrt{\frac{6\pi\sqrt{3}}{4\pi+3\sqrt{3}}}$	$2\sqrt{\frac{2\pi\sqrt{3}}{4\pi+3\sqrt{3}}}$	$\frac{4\pi+3\sqrt{3}}{6}$

For the Werenskiold III is the same Wagner IV but with the C_x and C_y values are increased by 1.15862.

5.2.15 Hölzel.

`+proj=holzel` Fig. 5.5

$$x = \lambda \begin{cases} .322673 + .369722\left[1 - \left(\frac{|\phi| - .40928}{1.161517}\right)^2\right]^{\frac{1}{2}} & \text{if } |\phi| > 1.39634 \\ .441013 * (1 + \cos\phi) & \text{otherwise} \end{cases} \tag{5.41}$$
$$y = \phi \tag{5.42}$$

5.2.16 Hatano.

`+proj=hatano [+sym]` Fig. 5.5 Ref. [17, p. 64 and 221]

If the option `+syn` is selected, the symmetric form of this projection is used, otherwise the asymmetric form.

$$x = 0.85\lambda\cos\theta \tag{5.43}$$
$$y = C_y \sin\theta \tag{5.44}$$
$$P(\theta) = 2\theta + \sin 2\theta - C_p \sin\phi \tag{5.45}$$
$$P'(\theta) = 2(1 + \cos 2\theta) \tag{5.46}$$
$$\theta_0 = 2\phi \tag{5.47}$$

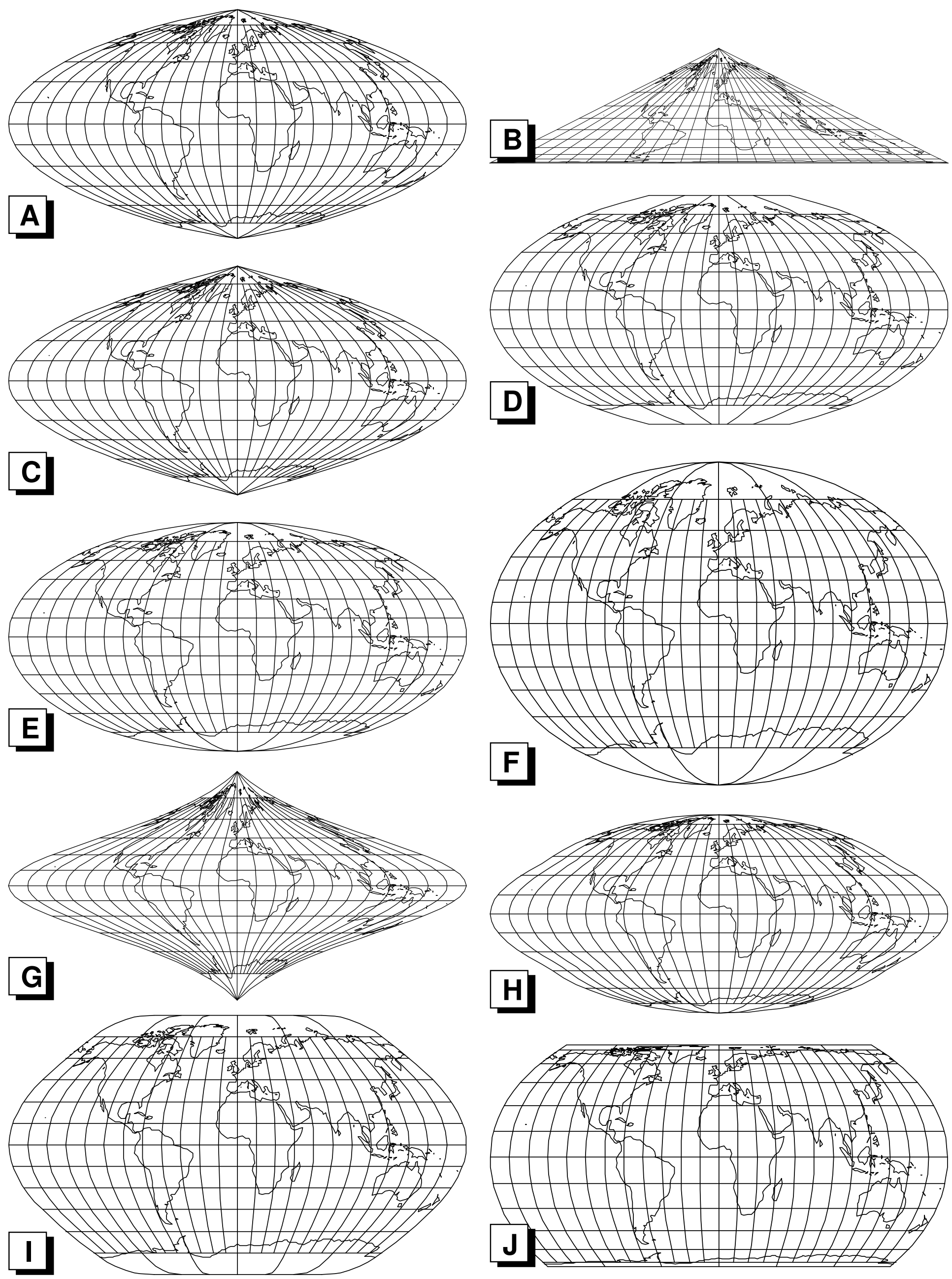

Figure 5.5: General pseudocylindricals II
A–Boggs Eumorphic, **B**–Collignon, **C**–Craster, **D**–Denoyer, **E**–Equidistant Mollweide, **F**–Fahey, **G**–Foucault, **H**–Goode Homolosine, **I**–Hölzel and **J**–Hatano.

	C_y	C_p
if `+sym` or $\phi > 0$	1.75859	2.67595
if not `+sym` and $\phi < 0$	1.93052	2.43763

For $\phi = 0$, $y \leftarrow 0$ and $x \leftarrow 0.85\lambda$.

5.2.17 Craster (Putniņš P_4).

`+proj=crast` Fig. 5.5 Ref. [? , p. 221]

A pointed pole, equal-area projection with standard parallels at 36°46′.

$$x = \sqrt{3/\pi}\lambda[2\cos(2\phi/3) - 1] \qquad y = \sqrt{3\pi}\sin(\phi/3) \tag{5.48}$$

5.2.18 Putniņš P_2.

`+proj=put2` Fig. 5.6 Ref. [13, p.66]

$$x = 1.89490\lambda(\cos\theta - 1/2) \tag{5.49}$$
$$y = 1.71848\sin\theta \tag{5.50}$$
$$P(\theta) = 2\theta + \sin 2\theta - 2\sin\theta - [(4\pi - 3\sqrt{3})/6]\sin\phi \tag{5.51}$$
$$= \theta + \sin\theta(\cos\theta - 1) - [(4\pi - 3\sqrt{3})/12]\sin\phi$$
$$P'(\theta) = 2 + 2\cos 2\theta + 2\cos\theta \tag{5.52}$$
$$= 1 + \cos\theta(\cos\theta - 1) - \sin^2\theta$$
$$\theta_0 = 0.615709\phi + 0.00909953\phi^3 + 0.0046292\phi^5 \tag{5.53}$$

5.2.19 Putniņš P_3 and P'_3.

Name	+proj=	figure	Ref.
Putniņš P_3	putp3	5.6	[13, p. 69]
Putniņš P'_3	putp3p	5.6	[13, p. 69]

$$x = \sqrt{2/\pi}\lambda(1 - A\phi^2/\pi^2) \qquad y = \sqrt{2/\pi}\phi \tag{5.54}$$

where A is 4 and 2 for respective P_3 and P'_3.

5.2.20 Putniņš P'_4 and Werenskiold I.

This is the flat pole version of Putniņš's P_4 or Craster's Parabolic.

Name	+proj=	figure	Ref.
Putniņš P_4	putp4	5.6	[13, p. 68]
Werenskiold I	weren	5.2	[13, p. 68]

$$x = C_x\lambda\cos\theta/\cos(\theta/3) \qquad y = C_y\sin(\theta/3) \tag{5.55}$$
$$\sin\theta = (5\sqrt{2}/8)\sin\phi \tag{5.56}$$

where

	P'_4	Weren. I
C_x	$2\sqrt{0.6/\pi}$	1.0
C_y	$2\sqrt{1.2\pi}$	$\pi\sqrt{2}$

5.2.21 Putniņš P_5 and P'_5.

Putniņš P_5 and P'_5 projections have equally spaced parallels and respectively pointed and flat poles.

Name	+proj=	figure	Ref.
Putniņš P_5	putp5	5.6	[13, p. 69]
Putniņš P'_5	putp5p	5.6	[13, p. 69]

$$x = 1.01346\lambda(A - B\sqrt{1 + 12\phi^2/\pi^2}) \qquad y = 1.01346\phi \tag{5.57}$$

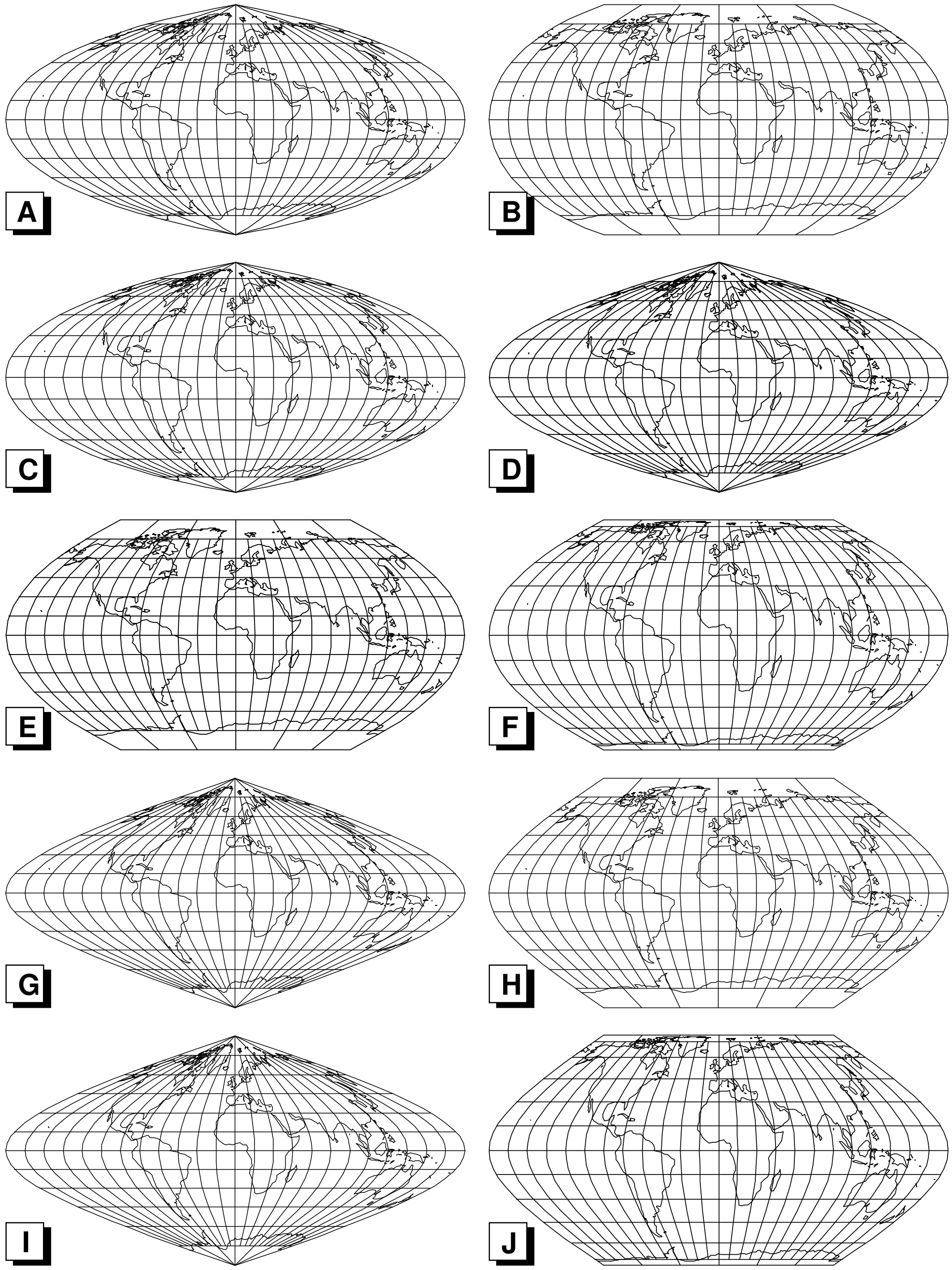

Figure 5.6: PutniṇšPseudocylindricals.
A–Putniṇš P_1, **B**–Putniṇš P_1', **C**–Putniṇš P_2, **D**–Putniṇš P_3, **E**–Putniṇš P_3', **F**–Putniṇš P_4', **G**–Putniṇš P_5, **H**–Putniṇš P_5', **I**–Putniṇš P_6 and **J**–Putniṇš P_6'.

where

	P_5	P_5'
A	2.0	1.5
B	1.0	0.5

5.2.22 Putniņš P_6 and P'_6.

Putniņš P_6 and P'_6 projections are equal-area with respective pointed and flat poles.

Name	+proj=	figure	Ref.
Putniņš P_6	`putp6`	5.6	[13, p. 69]
Putniņš P'_6	`putp6p`	5.6	[13, p. 69]

$$x = C_x\lambda(D - (1+p^2)^{1/2}) \tag{5.58}$$
$$y = C_y p \tag{5.59}$$
$$P(p) = (A - (1+p^2)^{1/2})p - \ln(p + (1+p^2)^{1/2}) - B\sin\phi \tag{5.60}$$
$$P'(p) = A - 2\sqrt{1+p^2} \tag{5.61}$$
$$p_0 = \phi \tag{5.62}$$

where

	P_6	P'_6
C_x	1.01346	0.44329
D	2	3
C_y	0.91910	0.80404
A	4.00000	6.00000
B	2.14714	5.61125

5.2.23 Collignon.

`+proj=collg` Fig. 5.5 [17, p. 223]

$$x = (2/\sqrt{\pi})\lambda\sqrt{1-\sin\phi} \qquad y = \sqrt{\pi}(1 - \sqrt{1-\sin\phi}) \tag{5.63}$$

5.2.24 Sine-Tangent Series.

Name	+proj=	figure	Ref.
Foucaut	`fouc`	5.5	[13, p. 70]
Adams Quartic Authalic	`qua_aut`	5.9	[13, p. 70]
McBryde-Thomas Sine (No. 1)	`mbt_s`	5.7	[13, p. 72]
Kavraisky V	`kav5`	5.2	[13, p. 72]
General Sine/Tan.	`gen_ts [+t|+s] +q= +p=`		

Baar [**?**] listed several variations with values of $p = q = 10/9$, $4/3$ and $3/2$ for the sine series and 1, 4/3, 3/2 and 3 for the tangent series.

Sine seriesi equations:

$$x = (q/p)\lambda\cos\phi/\cos(\phi/q) \qquad y = p\sin(\phi/q) \tag{5.64}$$

Tangent Seriesi equations:

$$x = (q/p)\lambda\cos\phi\cos^2(\phi/q) \qquad y = p\tan(\phi/q) \tag{5.65}$$

q	p	Sine	Tangent
2	$\sqrt{\pi}$		Foucaut
2	2	Quartic Authalic	
1.36509	1.48875	McBryde-Thomas	
35° arccos(0.9)	$q/0.9$	Kavraisky V	

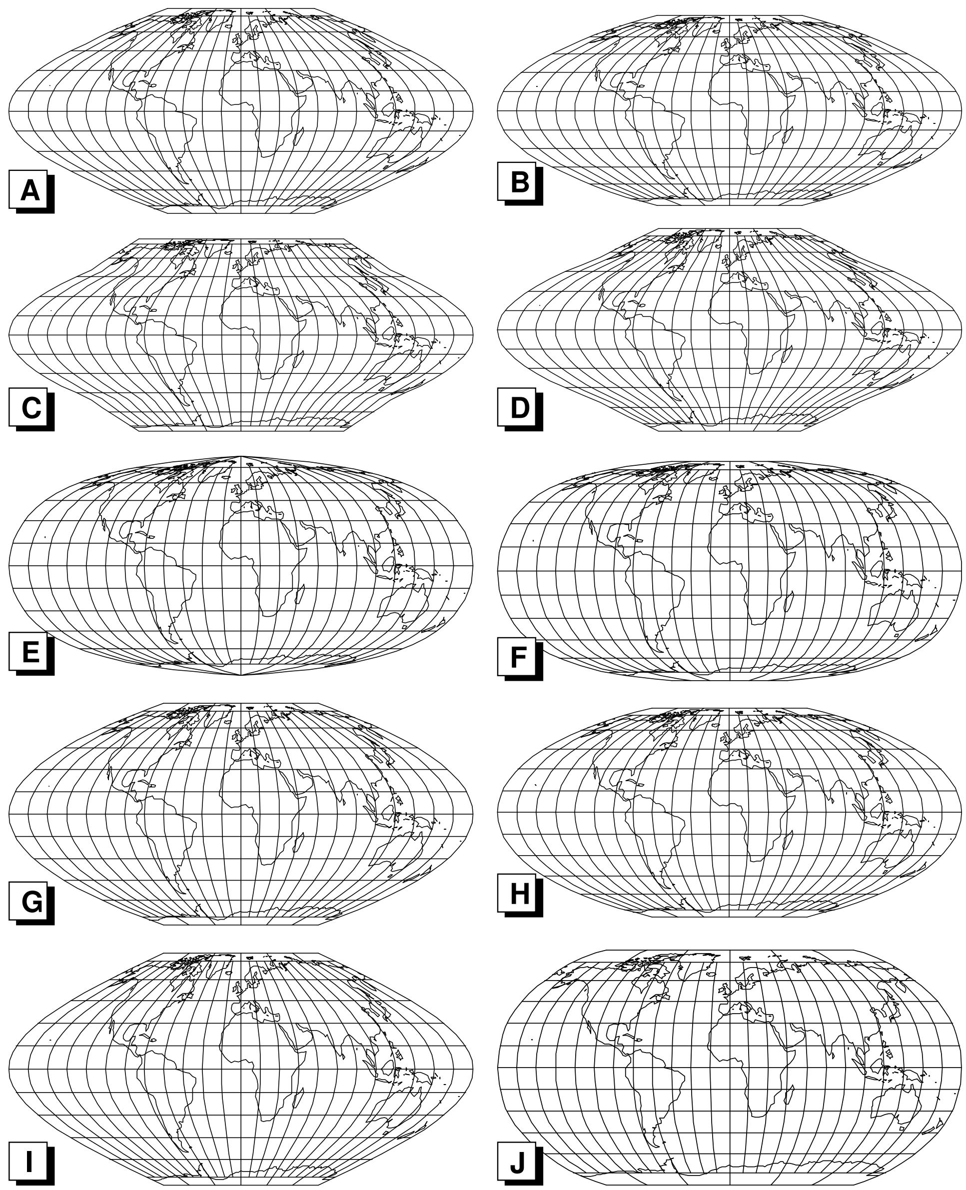

Figure 5.7: General pseudocylindricals III
A–McBryde P3, **B**–McBryde Q3, **C**–McBryde S2, **D**–McBryde S3, **E**–McBryde-Thomas Sine (No. 1), **F**–McBryde-Thomas Flat-Poler Sine (No. 2) **G**–McBride-Thomas Flat-Polar Parabolic, **H**–McBryde-Thomas Flat-Polar Quartic, **I**–McBryde-Thomas Flat-Polar Sinusoidal and **J**–Robinson .

5.2.25 McBryde-Thomas Flat-Polar Parabolic.

`+proj=mbtfpp` Fig. 5.7

$$x = \sqrt{6/7}/3\lambda[1 + 2\cos\theta/\cos(\theta/3)] \tag{5.66}$$
$$y = 3\sqrt{6/7}\sin(\theta/3) \tag{5.67}$$
$$P(\theta) = 1.125\sin(\theta/3) - \sin^3(\theta/3) - 0.4375\sin\phi \tag{5.68}$$
$$P'(\theta) = [0.375 - \sin^2(\theta/3)]\cos(\theta/3) \tag{5.69}$$
$$\theta_0 = \phi \tag{5.70}$$

5.2.26 McBryde-Thomas Flat-Polar Sine (No. 1).

`+proj=mbtfps` Fig. 5.7

$$x = 0.22248\lambda[1 + 3\cos\theta/\cos(\theta/1.36509)] \tag{5.71}$$
$$y = 1.44492\sin(\theta/1.36509) \tag{5.72}$$
$$P(\theta) = 0.45503\sin(\theta/1.36509) + \sin\theta - 1.41546\sin\phi \tag{5.73}$$
$$P'(\theta) = \frac{0.45503}{1.36509}\cos(\theta/1.36509) + \cos\theta \tag{5.74}$$
$$\theta = \phi \tag{5.75}$$

At the moment, there is a discrepancy between formulary and claim that 80° parallel length is half the length of the equator.

5.2.27 McBryde-Thomas Flat-Polar Quartic.

`+proj=mbtfpq` Fig. 5.7

$$x = \lambda(1 + 2\cos\theta/\cos(\theta/2))[3\sqrt{2} + 6]^{-1/2} \tag{5.76}$$
$$y = (2\sqrt{3}\sin(\theta/2)[2 + \sqrt{2}]^{-1/2} \tag{5.77}$$
$$P(\theta) = \sin(\theta/2) + \sin\theta - (1 + \sqrt{2}/2)\sin\phi \tag{5.78}$$
$$P'(\theta) = (1/2)\cos(\theta/2) + \cos\theta \tag{5.79}$$
$$\theta = \phi \tag{5.80}$$

5.2.28 Boggs Eumorphic.

`+proj=boggs` Fig. 5.5

$$x = 2.00276\lambda(\sec\phi + 1.11072\sec\theta) \qquad y = 0.49931(\phi + \sqrt{2}\sin\theta) \tag{5.81}$$
$$P(\theta) = 2\theta + \sin 2\theta - \pi\sin\phi \qquad P'(\theta) = 2 + 2\cos 2\theta \tag{5.82}$$
$$\theta = \phi \tag{5.83}$$

5.2.29 Nell.

`+proj=nell` Fig. 5.8 Ref. [15][p. 115]

$$x = \lambda(1 + \cos\theta)/2 \qquad y = \theta \tag{5.84}$$
$$P(\theta) = \theta + \sin\theta - 2\sin\phi \qquad P'(\theta) = 1 + \cos\theta \tag{5.85}$$
$$\theta_1 = 1.00371\phi - 0.0935382\phi^3 - 0.011412\phi^5 \tag{5.86}$$

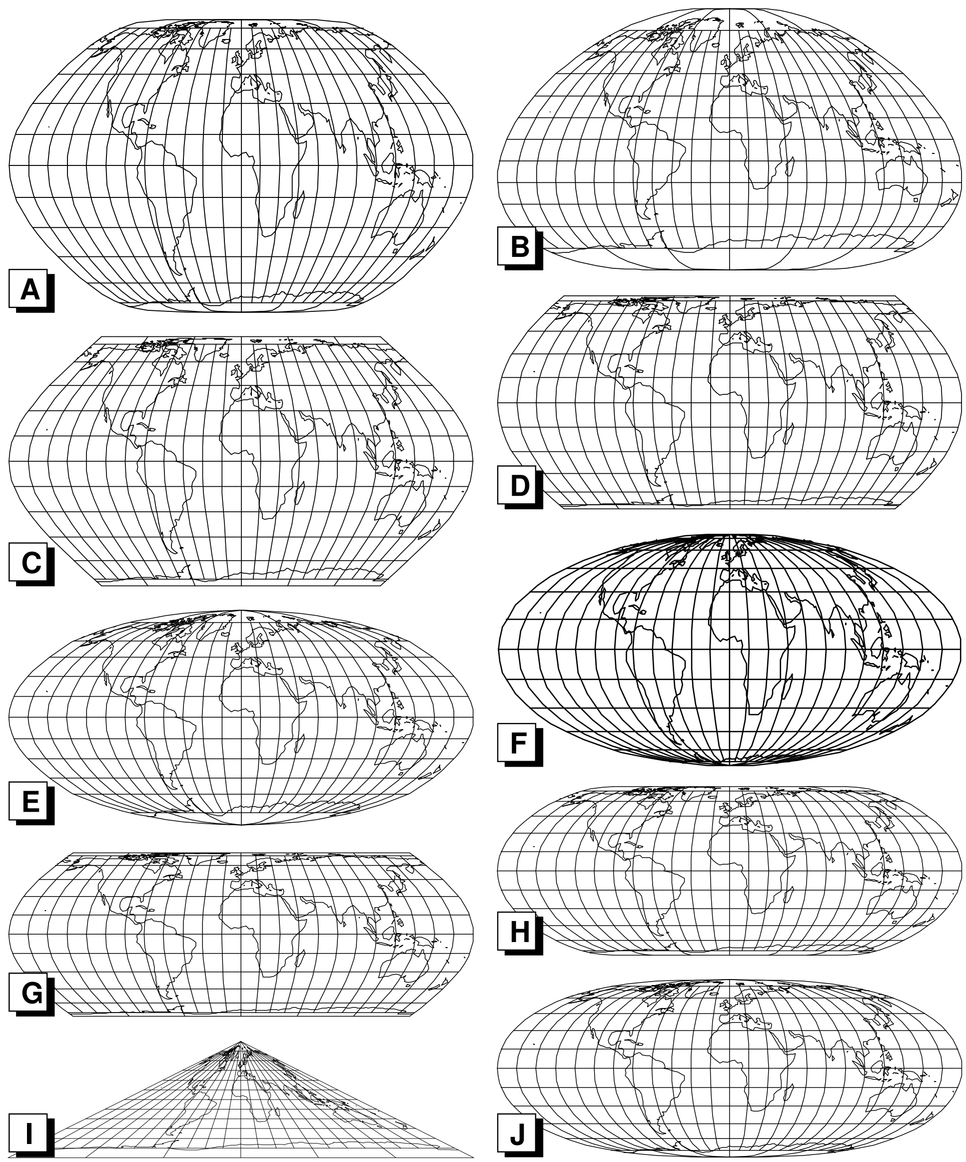

Figure 5.8: General pseudocylindricals IV
A–Snyder Minimum Error. **B**–Loximuthal (`+lat_1=40`), **C**–Urmayev V, **D**–Urmayev Flat-Polar Sinusoidal (`+n=0.7`), **E**–Érdi-Krausz, **F**–Fourtier II, **G**–Nell–Hammer, **H**–Nell, **I**–Maurer and **J**–Mayr–Tobler.

5.2.30 Nell-Hammer.

`+proj=nell_h [+n=]` Fig. 5.8 Ref. [13, p. 74]
The equal-area Nell-Hammer is a specialized case of the more generalized arithmetic mean of tha y-axis or parallels of the Cylindical Equal-Area and the Sinusoidal

projection [19]:

$$x = (a + b\cos\phi)\lambda \tag{5.87}$$

$$y = \begin{cases} 2\left(\phi - \tan\frac{\phi}{2}\right) & \text{for } a = b = 1/2 \\ \dfrac{\phi}{b} - \dfrac{a}{b}\begin{cases} \dfrac{2}{\sqrt{a^2-b^2}}\arctan\dfrac{\sqrt{a^2-b^2}\tan\frac{\phi}{2}}{a+b} & \text{if } a^2 > b^2 \\ \dfrac{2}{\sqrt{b^2-a^2}}\operatorname{arctanh}\dfrac{(b-a)\tan\frac{\phi}{2}}{\sqrt{b^2-a^2}} & \text{if } b^2 > a^2 \end{cases} \end{cases} \tag{5.88}$$

where a and b are the respective weights of the cylindrical equal-area and sinuisoidal projections and where $a + b = 1$.

The optional `n` parameter corresponds to a and $0 < n < 1$. When `n` is not specified then $n \leftarrow 0.5$ (true Nell-Hammer).

5.2.31 Robinson.

`+proj=robin` Fig. 5.7 Ref. [11]
Common for global thematic maps in recent atlases.

$$x = 0.8487\lambda X(|\phi|) \qquad y = 1.3523Y(|\phi|) \quad y \text{ assumes sign of } \phi \tag{5.89}$$

where the coefficients of X and Y are determined from the following table:

$\phi°$	Y	X	$\phi°$	Y	X
0	0.0000	1.0000	50	0.6176	0.8679
5	0.0620	0.9986	55	0.6769	0.8350
10	0.1240	0.9954	60	0.7346	0.7986
15	0.1860	0.9900	65	0.7903	0.7597
20	0.2480	0.9822	70	0.8435	0.7186
25	0.3100	0.9730	75	0.8936	0.6732
30	0.3720	0.9600	80	0.9394	0.6213
35	0.4340	0.9427	85	0.9761	0.5722
40	0.4968	0.9216	90	1.0000	0.5322
45	0.5571	0.8962			

Robinson did not define how intermediate values were to be interpolated between the 5° intervals. The **proj** system uses a set of bicubic splines determined for each X–Y set with zero second derivatives at the poles. GCTP uses Stirling's interpolation with second differences.

5.2.32 Denoyer.

`+proj=denoy` Fig. 5.5

$$x = \lambda\cos[(0.95 - \lambda/12 + \lambda^3/600)\phi] \qquad y = \phi \tag{5.90}$$

5.2.33 Fahey.

`+proj=fahey` Fig. 5.5

$$x = \lambda\cos 35°\sqrt{1 - \tan^2(\phi/2)} \qquad y = (1 + \cos 35°)\tan(\phi/2) \tag{5.91}$$

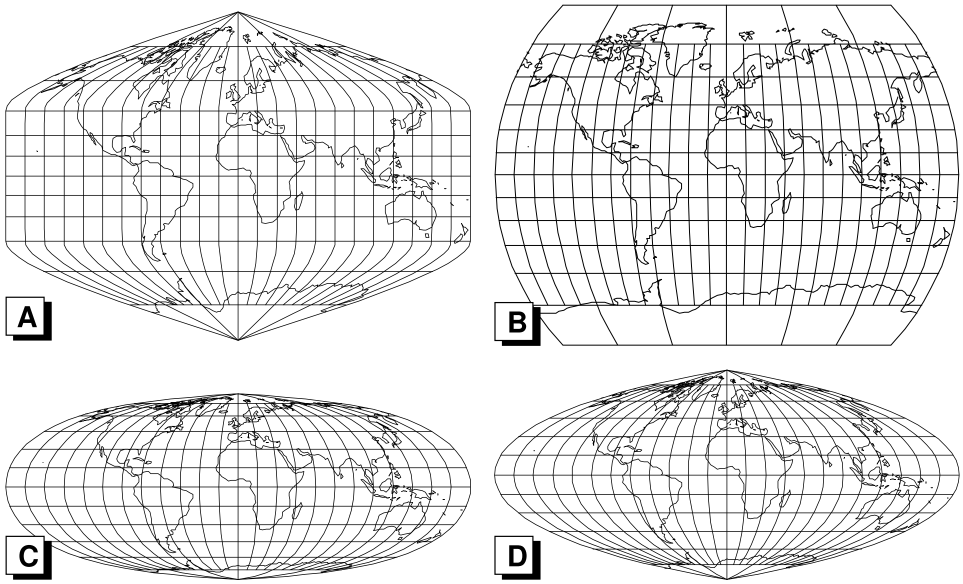

Figure 5.9: General pseudocylindricals V
A–Baker Dinomic, **B**–Times, **C**–Tobler G1 and **D**–Quartic Authalic.

5.2.34 Ginsburg VIII.

`+proj=gins8` Fig. 5.10 [13][p. 78]

$$x = \lambda(1 - 0.162388\phi^2)(0.87 - 0.000952426\lambda^4) \qquad y = \phi(1 + \phi^3/12) \tag{5.92}$$

5.2.35 Loximuthal.

`+proj=loxim +lat_1=` Fig. 5.8
All straight lines radiating from the point where `lat_1=`ϕ_1 intersects the central meridian are loxodromes (rhumb lines) and scale along the loxodomes is true.

$$x = \begin{cases} \lambda(\phi - \phi_1)/[\ln\tan(\pi/4 + \phi/2) - \ln\tan(\pi/4 + \phi_1/2)] & \phi \neq \phi_1 \\ \lambda\cos\phi_1 & \phi = \phi_1 \end{cases} y \quad = \phi - \phi_1 \tag{5.93}$$

5.2.36 Urmayev V Series.

`+proj=urm5` Fig. 5.8 Ref. [13, p. 77] [15][213]

$$x = m\lambda\cos\theta \tag{5.94}$$
$$y = \theta(1 + q\theta^2/3)/(mn) \tag{5.95}$$
$$\sin\theta = n\sin\phi \tag{5.96}$$

where $m = 2\sqrt[4]{3}/3$, $n = 0.8$ and $q = 0.414524$ are default values that have been employed in some atlases.

5.2.37 Goode Homolosine, McBryde Q3 and McBride S2.

Name	+proj=	figure	Ref.
Goode Homolosine	`goode`	5.5	
McBryde P3	`psfig:mb_P3`	5.7	
McBryde Q3	`psfig:mb_Q3`	5.7	
McBryde S2	`psfig:mb_S2`	5.7	

Pseudocylindrical can be composited where different projections are used in different latitude zones. In the cases presented here there are only two regions: one covering the central or equitorial latitudes and another covering the polar regions. At the latitude where they join together, the horizontal scale must match and a shift value is normally subtracted from the computed y-value of the polar projection.

Name	Equitorial	Polar	ϕ join	y offset
Goode Homolosine	Sinusoidal Sec. 5.2.1	Mollweide Sec 5.2.14	$40°44'$	0.05280
McBryde P3	Craster Parabolic Sec. 5.2.17	McBryde-Thomas Flat-Polar Parabolic Sec. 5.2.25	$49°20'21.8''$	0.035509
McBryde Q3	Quartic Authalic Sec. 5.2.24	McBryde-Thomas Flat-Polar Quartic Sec. 5.2.27	$52°9'$	0.042686
McBryde S2	Sinusoidal	Eckert VI Sec. 5.2.8	$49°16'$	0.084398

5.2.38 Equidistant Mollweide

`+proj=eq_moll` Fig. 5.5

$$x = \frac{\lambda}{\pi}\sqrt{|\pi^2 - 4\phi^2|} \qquad y = \phi \tag{5.97}$$

5.2.39 McBryde S3.

`+proj=mb_S3` Fig. 5.7 Ref.

If $|\phi| < 55°51'$ then

$$x = \lambda\cos\phi \qquad y = \phi \tag{5.98}$$

else

$$x = \frac{C\lambda}{1.5}(0.5 + \cos\theta) \tag{5.99}$$

$$y = C\theta \mp 0.069065 \tag{5.100}$$

$$P(\theta) = \frac{\theta}{2} + \sin\theta - (1 - \frac{\pi}{4})\sin\phi \tag{5.101}$$

$$P'(\theta) = \frac{1}{2} + \cos\theta \tag{5.102}$$

$$\theta_0 = \phi \tag{5.103}$$

$$C = \left(\frac{6}{4+\pi}\right)^{\frac{1}{2}} \tag{5.104}$$

$$\tag{5.105}$$

where the last constant takes the opposite sign of ϕ.

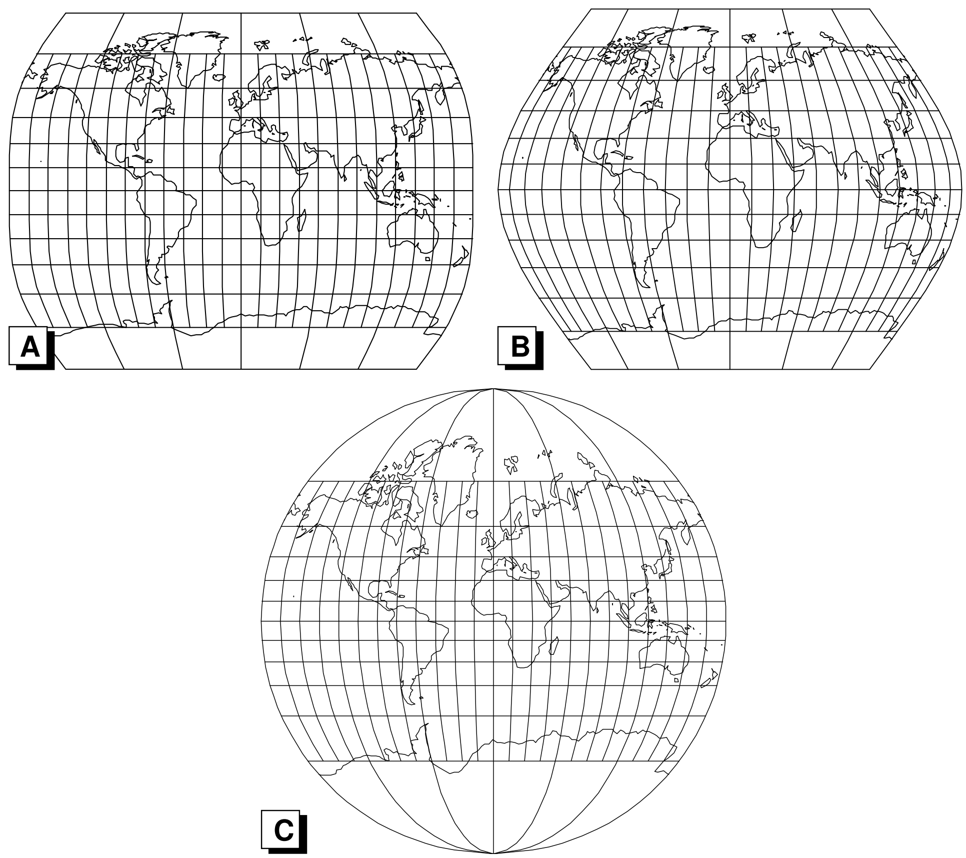

Figure 5.10: General pseudocylindricals VI
A–Oxford, **B**–Ginsburg VIII and **C**–Semiconformal.

5.2.40 Semiconformal.

`+proj=near_con` (Fig. 5.10

$$p = \begin{cases} \text{sign of } \phi \; 0.99989 & \text{if } |\phi| > 1.5564 \\ \sin\phi & \text{otherwise} \end{cases} \tag{5.106}$$

$$\theta = \frac{1}{2\pi} \ln\left(\frac{1+p}{1-p}\right) \tag{5.107}$$

$$x = \lambda\cos\theta \tag{5.108}$$

$$y = \pi\sin\theta \tag{5.109}$$

5.2.41 Érdi-Krausz.

`+proj=erdi_krusz` Fig. 5.8 Ref. [13, p. 73–74]
If $|\phi| < \pi/3$ then

$$x = 0.96042\lambda\cos\theta' \qquad y = 1.30152\theta' \tag{5.110}$$

$$\sin\theta' = 0.8\sin\phi \tag{5.111}$$

otherwise

$$x = 1.07023\lambda\cos\theta \qquad y = 1.68111\sin\theta \mp 0.28549 \tag{5.112}$$

where sign is opposite that of θ

$$P(\theta) = 2\theta + \sin 2\theta - \pi \sin\phi \qquad P'(\theta) = 2 + 2\cos 2\theta \tag{5.113}$$
$$\theta_0 = \phi \tag{5.114}$$

5.2.42 Snyder Minimum Error.

`+proj=s_min_err` Fig. 5.8

$$a_1 = 1.27326 \tag{5.115}$$
$$a_3 = -.04222 \tag{5.116}$$
$$a_5 = -.0293 \tag{5.117}$$
$$a'_3 = -0.12666 \tag{5.118}$$
$$a'_5 = -.1465 \tag{5.119}$$
$$x = \frac{\lambda\cos\phi}{a_1 + \phi^2(a'_3 + a'_5\phi^2)} \tag{5.120}$$
$$y = \phi(a_1 + \phi^2(a_3 + a_5\phi^2)) \tag{5.121}$$

5.2.43 Maurer.

`+proj=maurer` Fig. 5.8 [13, p. 69]

$$x = \lambda\left(\frac{\pi - 2\phi}{\pi}\right) \qquad y = \phi \tag{5.122}$$

5.2.44 Canters.

Canters' four low-error pseudocylindrical projections.

Name	`+proj=`	figure	Ref
General optimization	`fc_gen`	5.11 for all	[6]
Pole length half the lenght of the equator	`fc_pe`		
Correct axis ratio	`fc_ar`		
Pointed pole, correct axis ratio	`fc_pp`		

with the general form:

$$x = \lambda(c_0 + c_2\phi^2 + c_4\phi^4)\begin{cases}\text{flat polar} \\ \times\cos\phi \quad \text{pointed pole}\end{cases} \tag{5.123}$$
$$y = c'_1\phi + c'_3\phi^3 + c'_5\phi^5 \tag{5.124}$$

where the coefficients are:

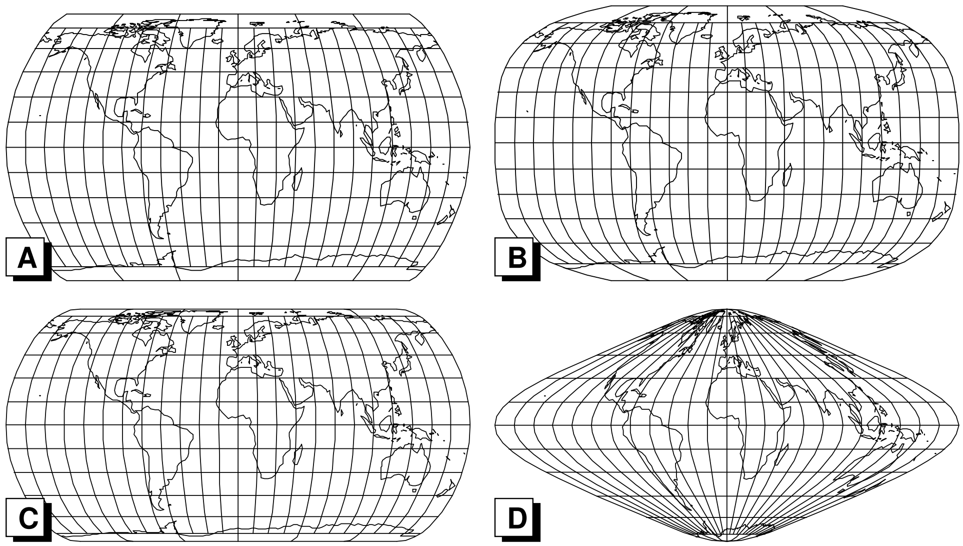

Figure 5.11: Canters' pseudocylindrical series
A–Canters' General optimization, **B**–Pole length half the lenght of the equator. **C**–Pole length half the lenght of the equator and **D**–Pointed pole, correct axis ratio.

general optimization	
$c_0 = 0.7920$	$c'_1 = 1.0304$
$c_2 = -0.0978$	$c'_3 = 0.0127$
$c_4 = 0.0059$	$c'_5 = -0.0250$
pole length half the lenght of the equator	
$c_0 = 0.7879$	$c'_1 = 1.0370$
$c_2 = -0.0238$	$c'_3 = -0.0059$
$c_4 = -0.0551$	$c'_5 = -0.0147$
correct axis ratio and	
$c_0 = 0.8378$	$c'_1 = 1.0150$
$c_2 = -0.1053$	$c'_3 = 0.0207$
$c_4 = -0.0011$	$c'_5 = -0.0375$
pointed pole, correct axis ratio	
$c_0 = 0.8333$	$c'_1 = 1.0114$
$c_2 = 0.3385$	$c'_3 = 0.0243$
$c_4 = 0.0942$	$c'_5 = -0.0391$

5.2.45 Baranyi I–VII.

Name	`proj=`	figure
Baranyi IV (Snyder)	`baranyi4`	
Baranyi I	`brny_1 +vopt`	all on fig. 5.12
Baranyi II	`brny_2 +vopt`	
Baranyi III	`brny_3`	
Baranyi IV	`brny_4`	
Baranyi V	`brny_5`	
Baranyi VI	`brny_6`	
Baranyi VII	`brny_7`	

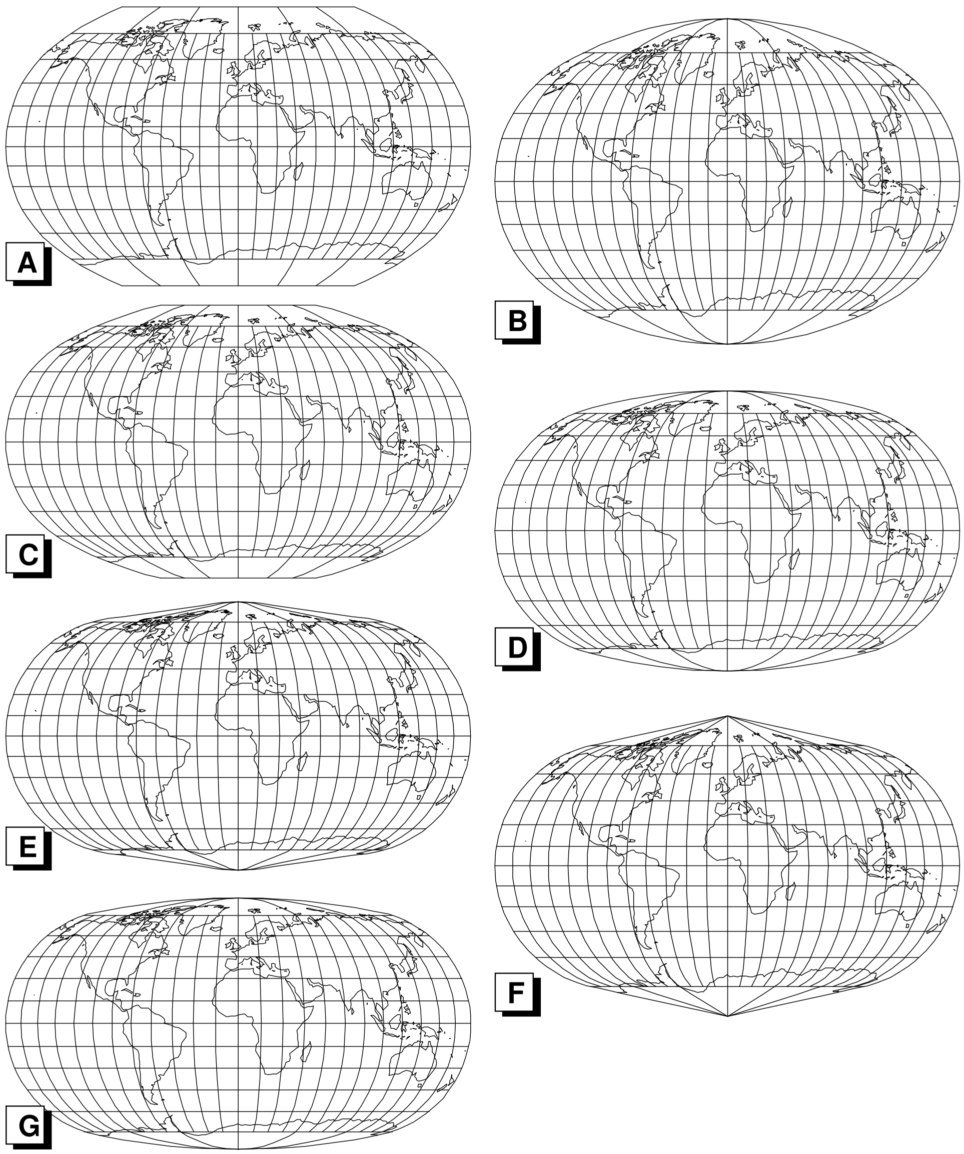

Figure 5.12: Baranyi pseudocylindrical series
A–Baranyi I, **B**–II, **C**–III, **D**–IV **E**–V **F**–VI and **G**–VII.

Baranyi IV.

The following is a version of projection IV of the Baranyi set of seven projections [5] that is derived from unpublished BASIC procedure written by Snyder and forwarded by Anderson [4]:

$$y = \phi(1. + \phi^2(.112579 + |\phi|(-.107505 + |\phi|.0273759))) \tag{5.125}$$

$$f = \pm\frac{\log(1. + 0.11679 * |\lambda|)}{0.31255} \text{ where } f \text{ takes the sign of } \lambda$$

$$x = f \times \begin{cases} (1.22172 + \sqrt{2.115292 - y^2}) & \text{when } |\phi| \le 1.36258 \\ \sqrt{|38.4304449 - (4.5848 + |y|)^2|} & \text{otherwise} \end{cases} \tag{5.126}$$

Baranyi projections.

The following version of Baranyi's seven projections attempt to stay close to Baranyi's original description [5] with some interpretations from a FORTRAN procedure by Voxland [9].

The projections follow three basic steps:

- convert both latitude and longitude to intermediate units (x_p, y_p) by means of tabular description of the converted coordinates at 10° intervals,
- determine the length of the parallel (x_l) for the intermediate latitude at the limiting longitude (180°) and
- scale the intermediate longitude by the ratio of the meridian length at the intermediate latitude and equatorial length.

Conversion if longitude and latitude to intermediate units is performed by first changing radians to degrees and then interpolating intermediate values from the from tables 5.1 and 5.2 of intermediates values at each $10^c irc$ of geographic coordinate. Because projections I and II have regular spacing or increments of tabular values, Voxland used a second degree determination for intermediate latitude:

$$y_p = a_1|\phi_d| + a_2\phi_d^2 \tag{5.127}$$

$$a_1 = \begin{cases} 0.975 & \text{Baranyi I} \\ 0.95 & \text{Baranyi II} \end{cases} \qquad a_2 = \begin{cases} 0.0025 & \text{Baranyi I} \\ 0.005 & \text{Baranyi II} \end{cases} \tag{5.128}$$

The results from the above equations for y_p will differ from the linear interpolation and may be selected by using the `+vopt` option. Although this solution is elegant it does not match the general nature of Baranyi's definition of the projection.

The previously determined intermediate longitude represent the value at the equator and must be scaled by the ratio of the length of the parallel at the intermediate latitude and length of the equator. Length of the parallels are determined by two or three segments that are either circular arcs or straight lines. Each segment joins in a smooth manner by the curves intersecting at points of tangency.

$$x_p = \frac{x_p}{x_p[180]} \begin{cases} X + \sqrt{R^2 - (y_p + Y)^2} & \text{circular arc} \\ (y_p - A)/B & \text{straight line segment} \end{cases} \tag{5.129}$$

$$\tag{5.130}$$

where the coefficients are determined from table 5.3. Applicable arc-line segment is determined by $y_p \leq y_p$-intersect column value. The empty last entry in this column is assumed to be infinite and thus selected if previous tests fail. Factor $x_p[180]$ is the length of the equator from the last column of table 5.2.

The intermediate coordinates are finally scaled to $x_p(10°)$ and $y_p(10°)$:

$$x = \pm x_p \frac{\pi}{180} \qquad y = \pm y_p \frac{\pi}{180} \tag{5.131}$$

where the sign of x and y are taken from λ and ϕ respectively.

	0°	$\lvert 10^\circ \rvert$	$\lvert 20^\circ \rvert$	$\lvert 30^\circ \rvert$	$\lvert 40^\circ \rvert$	$\lvert 50^\circ \rvert$	$\lvert 60^\circ \rvert$	$\lvert 70^\circ \rvert$	$\lvert 80^\circ \rvert$	$\lvert 90^\circ \rvert$
I	0.0	10.0	20.5	31.5	43.0	55.0	67.5	80.5	94.0	108.0
II	0.0	10.0	21.0	33.0	46.0	60.0	75.0	91.0	108.0	126.0
III	0.0	12.0	24.0	36.0	49.0	62.0	75.0	86.0	97.0	108.0
IV	0.0	12.0	24.0	36.0	49.0	62.0	75.0	87.0	99.0	111.0
V	0.0	10.0	20.5	31.5	44.0	58.0	70.5	81.5	92.0	102.0
VI	0.0	10.0	20.5	31.5	43.5	56.5	70.5	85.0	100.0	115.5
VII	0.0	12.0	24.0	35.5	47.0	58.5	69.5	80.5	90.5	99.5

Table 5.1: Intermediate parameter y_p value for each 10 degrees of latitude.

	0°	$\lvert 10^\circ \rvert$	$\lvert 20^\circ \rvert$	$\lvert 30^\circ \rvert$	$\lvert 40^\circ \rvert$	$\lvert 50^\circ \rvert$	$\lvert 60^\circ \rvert$	$\lvert 70^\circ \rvert$	$\lvert 80^\circ \rvert$	$\lvert 90^\circ \rvert$
	$\lvert 100^\circ \rvert$	$\lvert 110^\circ \rvert$	$\lvert 120^\circ \rvert$	$\lvert 130^\circ \rvert$	$\lvert 140^\circ \rvert$	$\lvert 150^\circ \rvert$	$\lvert 160^\circ \rvert$	$\lvert 170^\circ \rvert$	$\lvert 180^\circ \rvert$	
I	0.0	10.0	20.0	30.0	40.0	50.0	60.0	70.0	80.0	90.0
	100.0	110.0	120.0	130.0	140.0	150.0	160.0	170.0	180.0	
II	0.0	10.0	20.0	30.0	40.0	50.0	60.0	70.0	80.0	90.0
	100.0	110.0	120.0	130.0	140.0	150.0	160.0	170.0	180.0	
III	0.0	12.0	24.0	35.0	46.0	57.0	68.0	78.0	88.0	98.0
	108.0	118.0	128.0	138.0	148.0	157.0	166.0	175.0	184.0	
IV	0.0	12.0	24.0	35.0	46.0	57.0	68.0	78.0	88.0	98.0
	108.0	118.0	128.0	138.0	148.0	157.0	166.0	175.0	184.0	
V	0.0	10.5	21.0	31.5	42.0	52.5	62.5	72.5	82.5	92.5
	102.5	112.5	122.5	132.5	142.5	151.0	159.5	168.0	176.5	
VI	0.0	10.5	21.0	31.5	42.0	52.5	62.5	72.5	82.5	92.5
	102.5	112.5	122.5	132.5	142.5	151.5	160.5	169.5	178.5	
VII	0.0	12.0	24.0	35.5	47.0	58.0	69.0	79.5	90.0	100.0
	110.0	120.0	130.0	140.0	150.0	159.0	168.0	176.0	184.0	

Table 5.2: Intermediate parameter x_p value for each 10 degrees of longitude.

	Circular Arc [Line]			Arc–Line
No.	X	$Y[A]$	$R[B]$	y_p Intersect
I	80.0	0.0	100.0	81.241411756
	0.0	111.465034594	237.202237362	
II	75.0	0.0	105.0	89.732937686
	0.0	123.428571429	249.428571429	
III	94.0	0.0	90.0	78.300539425
	0.0	165.869652378	280.653459397	
IV	84.0	0.0	100.0	94.323113828
	0.0	315.227272727	426.227272727	
V	86.5	0.0	90.0	89.129742863
		[102.995921508]	[-0.140082858]	101.013708578
	0.0	0.0	102.0	
VI	83.5	0.0	95.0	92.807743792
		[115.5]	[-0.218634245]	
VII	94.0	0.0	90.0	87.968257449
	0.0	460.302631579	559.802631579	

Table 5.3: Table of limiting curve constants and y_p range limit.

5.2.46 Oxford and Times Atlas.

Name	+proj=	figure	Ref.
Oxford Atlas			
Modified Gall	`oxford`	5.10	
Times Atlas	`times`	5.9	

$$t = \tan\left(\frac{\phi}{2}\right) \tag{5.132}$$

$$y = \left(1 + \frac{\sqrt{2}}{2}\right) t \tag{5.133}$$

$$x = \lambda \begin{cases} \dfrac{1 - 0.04\phi^4}{\sqrt{2}} & \text{Oxford Atlas} \\ 0.74\sqrt{1 - 0.5t^2} & \text{Times Atlas} \end{cases} \tag{5.134}$$

5.2.47 Baker Dinomic.

`+proj=baker` Fig. 5.9 Ref. [15][p. 271]
When $|\phi| < \pi/4$ then projection is basic Mercator

$$x = \lambda \qquad y = \begin{cases} \ln\tan\left(\dfrac{\pi}{4} + \dfrac{\phi}{2}\right) & \text{or} \\ \dfrac{1}{2}\ln\left(\dfrac{1+\sin\phi}{1-\sin\phi}\right) \end{cases} \tag{5.135}$$

otherwise

$$x = \lambda\cos\phi\left(2\sqrt{2} - \csc\phi\right) \qquad y = \pm\left[-\ln\tan\frac{|\phi|}{2} + 2\sqrt{2}\left(|\phi| - \frac{\pi}{2}\right)\right] \tag{5.136}$$

where the above y value takes the sign of ϕ.

5.2.48 Fourtier II.

`+proj=four2` Fig. 5.8
A very early pseudocylindrical.

$$x = \frac{1}{\sqrt{\pi}}\cos\phi \qquad y = \frac{\sqrt{\pi}}{2}\sin\phi \tag{5.137}$$

5.2.49 Mayr-Tobler.

`+proj=mayr [+n=]` Fig. 5.8 Ref: [19], [15, p. 220]
An equal-area projection first described by Mayr and later by Tobler. The projection is based upon a weighted geometric mean of the x-axis or meridians of the Cylindical Equal-Area and Sinusoidal projections:

$$x = \lambda\cos^{1-n} b\phi \qquad y = \int_0^{\phi} \cos^n\phi\, d\phi \tag{5.138}$$

where $0 < n < 1$ and is the weight factor of the cylindrical projection. The Mayr projection is the special case (default) where $n \leftarrow 0.5$ when not specified.

5.2.50 Tobler G1

`+proj=tob_g1 [+n=]` Fig. 5.9 Ref. [19]

For this equal-area projection the y-axis is a weighted geometric mean of the Cylindrical Equal-Area and the Sinusoidal projections.

$$x = \lambda \cos\phi \frac{\phi^b \sin^a \phi}{a \sin\phi + b\phi \cos\phi} \tag{5.139}$$

$$y = \phi^a \sin^b \phi \tag{5.140}$$

$$a + b = 1 \tag{5.141}$$

Option `n` is equivalent to a and $0 < n < 1$ and is the weight for the cylindrical projection. If the `n` option is not specified, then $a \leftarrow 0.5$.

5.3 Pseudocylindrical Projections for the Ellipsoid.

5.3.1 Sinusoidal Projection

The elliptical version of the Sinusoidal projection is one of the simlplest elliptical computations. Spacing of the parallels is based upon the meridianal distance $M(\phi)$ as defined in section 3.2. The parallel lengths are determined by their radii defined in section 3.7.3. Thus the forward equations are simply:

$$x = a\lambda m(\phi) \qquad y = M(\phi) \tag{5.142}$$

The inverse projection values are determined by:

$$\phi = M^{-1}(y) \qquad \lambda = \frac{x}{m(\phi)} \tag{5.143}$$

Chapter 6

Conic Projections

Forward Conic Formulae.

The basic forward formulae for all simple conics are expressed by:

$$x = \rho \sin\theta \tag{6.1}$$
$$y = \rho_0 - \rho\cos\theta \tag{6.2}$$

where $\theta = n\lambda$. Factor ρ is the distance of the projected point from the apex of the cone and n is the cone constant. The factor ρ_0 is determined by evaluating ρ at ϕ_0 (`+lat_0=`) and establishes the y-axis origin. The x-axis origin is at λ_0 (`+lon_0=`).

Both ρ and n are functions that determine the characteristics of each conic projection, as shown in Table 6.1, and both usually controlled by two user specified parallels: ϕ_1 and ϕ_2 (`+lat_1=` and `+lat_2=`). In some cases, one parallel may be specified, ϕ_1, specified (in Lambert Equal Area and $\phi_1 = \phi_2$ cases) and in the case of the Lambert Conformal Conic, a scale factor, k_0 (`+k_0=`) may be specified. All cases where σ, σ or n would evaluate to 0 or n evaluates to 1 are not allowed.

In addition to the formulae for the spherical earth in Table 6.1 several conics are available for the ellipsoidal earth as follows.

Albers Equal Area:

$$\begin{aligned}
\rho &= \sqrt{C - nq}/n \\
n &= \begin{cases} (m_1^2 - m_2^2)/(q_2 - q_1) & \phi_1 \neq \phi_2 \\ \sin\phi_1 & \phi_1 = \phi_2 \end{cases} \\
C &= m_1^2 + nq_1 \\
m &= \cos\phi/(1 - e^2\sin^2\phi)^{1/2} \\
q &= (1 - e^2)\left[\frac{\sin\phi}{1 - e^2\sin^2\phi} - \frac{1}{2e} + \ln\left(\frac{1 - e\sin\phi}{1 + e\sin\phi}\right)\right] \\
k = 1/h &= \sqrt{C - nq}/m = n\rho/m
\end{aligned}$$

For the case of Lambert Equal Area, subsitute $\pi/2$ for ϕ_2 in the preceeding formulae.

Table 6.1: Spherical equations for conic projections.

Name	$\rho =$	$n =$
Equidistant	$\dfrac{\phi_2 \cos\phi_1 - \phi_1 \cos\phi_2}{\cos\phi_1 - \cos\phi_2} - \phi$	$\dfrac{\cos\phi_1 - \cos\phi_2}{\phi_2 - \phi_1}$
$\phi_1 = \phi_2$	$\cot\phi_1 + \phi_1 - \phi$	$\sin\phi_1$
Murdoch I	$(\cot\sigma \sin\delta)/\delta + \sigma - \phi$	$\sin\sigma$
Murdoch II	$\cot\sigma\sqrt{\cos\delta} + \tan(\sigma - \phi)$	$\sin\sigma\sqrt{\cos\delta}$
Murdock III	$\delta\cot\delta\cot\sigma + \sigma - \phi$	$(\sin\sigma\sin\delta\tan\delta)/\delta^2$
Euler	$\delta/2\cot(\delta/2)\cot\sigma + \sigma - \phi$	$(\sin\sigma\sin\delta)/\delta$
Lambert Conformal	$\cos\phi_1 \dfrac{\tan^n(\pi/4 + \phi_1/2)}{n\tan^n(\pi/4 + \phi/2)}$	$\dfrac{\ln\left(\dfrac{\cos\phi_1}{\cos\phi_2}\right)}{\ln\left(\dfrac{\tan(\pi/4+\phi_2/2)}{\tan(\pi/4+\phi_1/2)}\right)}$
$\phi_1 = \phi_2$	$k_0\cos\phi_1 \dfrac{\tan^n(\pi/4 + \phi_1)/2}{n\tan^n(\pi/4 + \phi/2)}$	$\sin\phi_1$
Albers Equal Area	$[\cos^2\phi_1 + 2n(\sin\phi_1 - \sin\phi)]^{1/2}/n$	$(\sin\phi_1 + \sin\phi_2)/2$
Lambert Equal Area	$[2(1 - \sin\phi)/n]^{1/2}$	$(1 + \sin\phi_1)/2$
Perspective	$\cos\delta[\cot\sigma - \tan(\phi - \sigma)]$	$\sin\sigma$
Tissot	$\left\{\left[\dfrac{\sin\sigma}{\cos\delta} + \dfrac{\cos\delta}{\sin\sigma} - 2\sin\phi\right]/n\right\}^{1/2}$	$\sin\sigma$
Vitkovsky I	same as Murdoch III	$(\tan\delta\sin\sigma)/\delta$

where $\sigma = (\phi_2 + \phi_1)/2$ and $\delta = (\phi_2 - \phi_1)/2$.

Lambert Conformal:

$$
\begin{aligned}
\rho &= k_0 F t^n \\
n &= \begin{cases} \dfrac{\ln m_1 - \ln m_2}{\ln t_1 - \ln t_2} & \phi_1 \neq \phi_2 \\ \sin\phi_1 & \phi_1 = \phi_2 \end{cases} \\
m &= \cos\phi/(1 - e^2 \sin^2\phi)^{1/2} \\
t &= \tan(\pi/4 - \phi/2) / \left[\frac{1 - e\sin\phi}{1 + e\sin\phi}\right]^{e/2} \\
&= \left[\left(\frac{1 - \sin\phi}{1 + \sin\phi}\right)\left(\frac{1 + e\sin\phi}{1 - e\sin\phi}\right)^e\right]^{1/2} \\
F &= m_1/(n t_1^n) \\
h = k &= k_0 n \rho / m \\
\gamma &= n\lambda
\end{aligned}
$$

Equidistant:

$$
\begin{aligned}
\rho &= G - \mathrm{M}(\phi) \\
n &= \begin{cases} (m_1 - m_2)/(\mathrm{M}(\phi_2) - \mathrm{M}(\phi_1)) & \phi_1 \neq \phi_2 \\ \sin\phi_1 & \phi_1 = \phi_2 \end{cases} \\
m &= \cos\phi/(1 - e^2 \sin^2\phi)^{1/2} \\
G &= m_1/n + \mathrm{M}(\phi_1) \\
h &= 1 \\
k &= n\rho/m
\end{aligned}
$$

Inverse Conic Formulae.

When $n < 0$, reverse the sign of x, y, and ϕ_0 and then evaluate: First compute $y' = \rho_0 - y$ and then

$$\rho = (x^2 + y'^2)^{1/2}.$$

If $n < 0$, reverse sign of x, y' and ρ and compute

$$\theta = \operatorname{atan2}(x, y')$$

Compute $\lambda = \theta/n$ and, in the spherical case, determine ϕ from the ρ equations in Table 6.1. Solutions for ϕ for the elliptical earth are as follows:

Albers Equal Area:

Compute $q = (C - \rho^2 n^2)/n$ and initial value of $\phi = \sin^{-1}(q/2)$ and iterate $\phi = \phi + \Delta\phi$ until $|\Delta\phi|$ less than acceptable tolerance and where:

$$\Delta\phi = \frac{(1 - e^e \sin^2)^{1/2}}{2\cos\phi}\left[\frac{q}{1 - e^2} - \frac{\sin\phi}{1 - e^2\sin^2\phi} + \frac{1}{2e}\ln\left(\frac{1 - e\sin\phi}{1 + e\sin\phi}\right)\right]$$

Lambert Conformal:

Compute $t = (\rho/F)^{1/n}$ and $\phi = \pi/2 - \tan^{-1} t$ and substitute ϕ into the right hand side of

$$\phi = \pi/2 - 2\tan^{-1}\left[t\left(\frac{1 - e\sin\phi}{1 + e\sin\phi}\right)^{e/2}\right]$$

Repeat substition of ϕ into right side until absolute difference between last and current value of ϕ less than tolerance.

Equidistant:

$\phi = \mathrm{M}^{-1}(G - \rho)$.

6.0.2 Bonne.

`+proj=bonne [+lat_1=]` **??** [14, p. 140]

The Werner and Sylvano are special cases of the Bonne where Werner specified a value of $\phi_1 = 90°$. Sylvano selected $\phi_1 = 47°$ and limited the geographic range to within $\pm 160°$ and $40°S \geq \phi \leq 80°N$ for a world map centered at $60°E$.

For forward of the sphere:

$$\rho = \cot \phi_1 + \phi_1 - \phi \tag{6.3}$$

$$E = \lambda \frac{\cos \phi}{\rho} \tag{6.4}$$

$$x = \rho \sin E \tag{6.5}$$

$$y = \cot \phi_1 - \rho \cos E \tag{6.6}$$

and for the ellipsoid:

$$m = \cos \phi / (1 - e^2 \sin^2 \phi)^{1/2} \tag{6.7}$$

$$\rho = m_1 / \sin \phi_1 + \mathrm{M}(\phi_1) - \mathrm{M}(\phi) \tag{6.8}$$

$$E = m\lambda / \rho \tag{6.9}$$

$$x = \rho \sin E \tag{6.10}$$

$$y = \frac{m_1}{\sin \phi_1} - \rho \cos E \tag{6.11}$$

For the inverse of the sphere:

$$\rho = \pm \left[x^2 + (\cot \phi_1 - y^2)^2\right]^{1/2} \tag{6.12}$$

$$\phi = \cot \phi_1 + \phi_1 - \rho \tag{6.13}$$

$$\lambda = \frac{\rho}{\cos \phi} \mathrm{atan2}[\pm x, \pm(\cot \phi_1 - y)] \tag{6.14}$$

and for the ellipsoid:

$$\rho = \pm \left[x^2 + (m_1 / \sin \phi_1 - y^2)^2\right]^{1/2} \tag{6.15}$$

$$\phi = \mathrm{M}^{-1}(m_1 / \sin \phi_1 + \mathrm{M}(\phi_1) - \rho) \tag{6.16}$$

$$\lambda = \frac{\rho}{m} \mathrm{atan2}[\pm x, \pm(m_1 / \sin \phi_1 - y)] \tag{6.17}$$

In all cases, $\pm$ take sign of ϕ_1.

6.0.3 Bipolar Oblique Conic Conformal.

Developed as a low-error conformal map of both North and South America, it consists of two translated, spherical form Lambert Conformal Conic projections (A and B) with points to the left of a geodesic from B to A determined by projection A and those to the right by projection B. There is a small and varying discontinuity along this geodesic, but it is negligible within the range of interest and at the small scales normally used. Because the formulae for this projection are presented by [14], they are used here rather than using a combination of the general oblique procedures. Only the spherical form is used and both `+lon_0` and `+lat_0` are ignored.

The following are defining constants for the location of the poles of each projection:

$$\begin{aligned}
\phi_A &= 20^\circ \text{ S} = -0.349065850398866 \\
\lambda_A &= 110^\circ \text{ W} = -1.91986217719376 \\
\phi_B &= 45^\circ \text{ N} = 0.785398163397448 \\
z_{AB} &= 104^\circ = 1.81514242207410
\end{aligned}$$

and each projection has the equivalent standard parallels of $\phi_1 = 31^\circ$ and $\phi_2 = 73^\circ$. These factors determine the following constants:

$$\begin{aligned}
\lambda_B &= \lambda_A + \arccos\left(\frac{\cos z_{AB} - \sin\phi_A \sin\phi_B}{\cos\phi_A \cos\phi_B}\right) \\
&= -0.348949767262507(-19^\circ 59' 36.0561) \\
n &= \ln\left(\frac{\sin\phi_1}{\sin\phi_2}\right) \ln\left(\frac{\tan(\phi_1/2)}{\tan(\phi_2/2)}\right) \\
&= 0.630558448812747 \\
F_0 &= \sin\phi_1/[n\tan^n(\phi_1/2)] \\
&= 1.83375966397205 \\
k_0 &= 2/[1 + nF_0 \tan^n 26^\circ / \sin 52^\circ] \\
&= 1.03462163714794 \\
F &= k_0 F_0 \\
&= 1.89724742567461 \\
Az_{AB} &= \arccos\{[\cos\phi_A \sin\phi_B - \\
&\quad \sin\phi_A \cos\phi_B \cos(\lambda_B + \lambda_A)]/\sin z_{AB}\} \\
&= 0.816500436746864(46^\circ 46' 55.30437'') \\
Az_{BA} &= \arccos\{[\cos\phi_B \sin\phi_A - \\
&\quad \sin\phi_B \cos\phi_A \cos(\lambda_B + \lambda_A)]/\sin z_{AB}\} \\
&= 1.82261843856186(104^\circ 25' 42.03909'') \\
T &= \tan^n(\phi_1/2) + \tan^n(\phi_2/2) \\
&= 1.27246578267089 \\
\rho_c &= FT/2 = 1.20709121521569 \\
z_c &= 2\tan^{-1}(T/2)^{1/n} \\
&= 0.908249725391265(52^\circ 2' 19.95363'')
\end{aligned}$$

Forward projection:

First determine which conic to use by computing

$$Az = \operatorname{atan2}[\sin(\lambda_B - \lambda),$$

$$\cos\phi_B \tan\phi - \sin\phi_B \cos(\lambda_B - \lambda)].$$

If $Az > Az_B A$, then conic A is to be used otherwise conic B. Next compute distance from point to conic pole from:

$$z = \arccos[\sin\phi_A \sin\phi + \cos\phi_A \cos\phi \cos(\lambda + \lambda_A)]$$

or

$$z = \arccos[\sin\phi_B \sin\phi + \sin\phi_B \cos\phi \cos(\lambda_B - \lambda)]$$

for respective A and B conics and in the case of the A conic recompute the azimuth from:

$$Az = \text{atan2}[(\sin(\lambda + \lambda_A),$$
$$\cos\phi_A \tan\phi - \sin\phi_A \cos(\lambda + \lambda_A)].$$

Next compute:

$$\begin{aligned} \rho &= F \tan^n(z/2) \\ k &= \rho n / \sin z \\ \alpha &= \arccos\{[\tan^n(z/2) + \tan^n(z_{AB} - z)]/T\} \end{aligned}$$

Determine θ by subtract Az from A_{AB} or A_{BA} for respective A or B conic and then compute

$$\rho = \rho / \cos[\alpha - \theta].$$

If $\rho > \alpha$, then $\rho = -\rho$. Now compute local cartesian system:

$$\begin{aligned} x' &= \rho \sin\theta \\ y' &= \mp\rho\cos\theta \pm \rho_c. \end{aligned}$$

using upper or lower sign for respective A or B conic. Finally, rotate into appropriate position:

$$\begin{aligned} x &= -x' \cos Az_c - y' \sin Az_c \\ y &= -y' \cos Az_c + x' \sin Az_c \end{aligned}$$

Inverse projection:

First, rotate cartesian into local system:

$$\begin{aligned} x' &= -x \cos Az_c + y \sin Az_c \\ y' &= -x \sin Az_c - y \cos Az_c \end{aligned}$$

If $x' < 0$, the change sign of y'. Next compute:

$$\begin{aligned} \rho &= [x'^2 + (\rho_c + y')]^{1/2} \\ A_z' &= \text{atan2}(x', \rho_c + y') \end{aligned}$$

Set $\rho = \rho'$ and compute

$$\begin{aligned} z &= 2\tan^{-1}(\rho/F)^{1/2} \\ \alpha &= \arccos\{[\tan^n(z/2) + \tan^n(z_{AB} - z)]\} \end{aligned}$$

If $Az < \alpha$, set $\rho = \rho'(\cos\alpha - Az)$ and repeat previous two equations until difference between previous and current ρ less than desired tolerance. If $x' < 0$, set $\phi_c = \phi_A$ and $A_c = Az_{AB}$, otherwise $\phi_c = \phi_B$ and $A_c = Az_{BA}$. Finally calculate:

$$\begin{aligned} Az &= A_c - Az/n \\ \phi &= \arcsin(\sin\phi_c \cos z + \cos\phi_c \sin z \cos Az \\ \lambda &= \lambda_c - \text{atan2}(\sin Az, \cos\phi_c / \tan z - \sin\phi_c \cos z) \end{aligned}$$

6.0.4 (American) Polyconic.

Forward:

If $\phi = 0$, then

$$x = \lambda$$
$$y = -m_0$$

otherwise

$$E = n\lambda \sin\phi$$
$$x = \cot\phi \sin E$$
$$y = m - m_0 + n\cot\phi(1 - \cos E)$$

For the sphere, $N = 1$ and $M = \phi$, and for the ellipsoid, $n = (1 - e^2 \sin^2\phi)^{-1/2}$ and $m = \mathrm{M}(\phi)$.

Inverse:

If $y = -m_0$, then

$$\phi = 0$$
$$\lambda = x$$

otherwise a Newton-Raphson approximation must be used which will not converge when $|\lambda| > \pi/2$. Firsts compute:

$$A = m_0 + y$$
$$B = x^2 + A^2$$

Set $\phi = A$ and iterate the following. For the sphere:

$$\Delta\phi = \frac{A(\phi\tan\phi + 1) - \phi - [(\phi^2 + B)\tan\phi]/2}{(\phi - A)/tan\phi - 1}$$

or the ellipsoid:

$$\begin{aligned}
C &= (1 - e^2\sin^2\phi)^{1/2}\tan\phi \\
m &= \mathrm{M}(\phi) \\
m' &= (1 - e^2)(1 - e^2\sin^2\phi)^{-3/2} \\
\Delta\phi &= [A(Cm + 1) - m - (M^2 + B)C/2]/ \\
&\quad [e^2 \sin 2\phi(m^2 + B - 2Am)/4C + \\
&\quad (A - m)(Cm' - 2/\sin 2\phi) - m']
\end{aligned}$$

For both: $\phi = \phi - \Delta\phi$ and recompute until $\Delta\phi$ less than tolerance. Lastly, compute $\lambda = \sin^{-1}(xC)/\sin\phi$.

IMW Polyconic.

A modified polyconic projection adopted in 1909 for the 1:1,000,000-scale International Map of the World where each panel spans 4° of latitude and with longitude extent determined by:

Latitude Zones	Longitude Range	λ_1
60°S to 60°N	6°	2°
60° to 76°	12°	4°
76° to 84°	24°	8°

The factor λ_1 is a *standard meridian* on either side of the map's central meridian that has unity meridianal scale factor (h). Parallel scale factor (k) is 1. along the map bounds. Longitude boundaries of standard IMW sheets are unknown to the author. Polar Stereographic was apparently used for the polar regions but confusing extent specifications make scale factor speculative.

Circa 1962, the Lambert Conformal Conic replaced this projection. For this system, the conic standard parallels are $1/5$ and $4/5$ of the extent of the 4° latitude zones (standard parallels obtained by adding by $\pm 48'$ respectively to lower and upper bounds) and the zones extend from 80°S to 84°N. The polar Stereographic projection is used for the remaining regions with scale factor adjusted to match the abutting edge of either IMW polyconic or Lambert Conformal Conic zones.

Usage of the IMW Polyconic requires specification of the map's limiting parallels, ϕ_1 and ϕ_2, with `lat_1` and `lat_2`. The projection may not symetrically span the equator ($\phi_1 = -\phi_2$). The standard merdians (from λ_1) are automatically determined from the mean of the standard parallels, but this my be overriden by specifying `lon_1`. Cartesian origin is at the central meridian, `lon_0`, and the most southerly bounding parallel.

Initializing computations for both forward and inverse are as follows. For $n = 1$, 2, compute

	$\phi_n \neq 0$	$\phi_n = 0$
$x_n =$	$R_n \sin F_n$	λ_1
$y_1 =$	$R_1(1 - \cos F_1)$	0
$T_2 =$	$R_2(1 - \cos F_2)$	0

where

$$\begin{aligned} R_n &= \cot \phi_n / (1 - e^2 \sin^2 \phi_n)^{1/2} \\ F_n &= \lambda_1 \sin \phi_n \end{aligned}$$

Then compute

$$\begin{aligned} y_2 &= \{[\mathrm{M}(\phi_2) - \mathrm{M}(\phi_1)]^2 - (x_2 - x_1)^2\}^{1/2} + y_1 \\ C_2 &= y_2 - T_2 \\ D &= \mathrm{M}(\phi_2) - \mathrm{M}(\phi_1) \\ P &= [\mathrm{M}(\phi_2) y_1 - \mathrm{M}(\phi_1) y_2]/D \\ Q &= (y_2 - y_1)/D \\ P' &= [\mathrm{M}(\phi_2) x_1 - \mathrm{M}(\phi_1) x_2]/D \\ Q' &= (x_2 - x_1)/D \end{aligned}$$

Forward:

Compute: *this is incomplete!!!* If $\phi = 0$, then

$$\begin{aligned} x &= \lambda \\ y &= 0 \end{aligned}$$

Otherwise, compute

$$\begin{aligned} x_a &= P' + Q' \mathrm{M}(\phi) \\ y_a &= P + Q \mathrm{M}(\phi) \\ R &= \cot \phi / (1 - e^2 \sin^2 \phi)^{1/2} \\ C &= y_a - R \pm (R^2 - x_a^2)^{1/2} \end{aligned}$$

where $\pm$ takes the same sign as ϕ. Next, compute

	$\phi_2 \neq 0$	$\phi_2 = 0$
$x_b =$	$R_2 \sin(\lambda \sin \phi_2)$	λ
$y_b =$	$C_2 + R_2[1 - \cos(\lambda \sin \phi_2)]$	C_2

and

	$\phi_1 \neq 0$	$\phi_1 = 0$
$x_c =$	$R_1 \sin(\lambda \sin \phi_1)$	λ
$y_c =$	$R_1[1 - \cos(\lambda \sin \phi_1)]$	0

Finally,

$$\begin{aligned} D &= (x_b - x_c)/(y_b - y_c) \\ B &= x_c + D(C + R - y_c) \\ x &= \{B \mp D[R^2(1 + D^2) - B^2]^{1/2}\}/(1 + D^2) \\ y &= C + R \mp (R^2 - x^2)^{1/2} \end{aligned}$$

where $\mp$ takes sign opposite of ϕ.

Inverse:

Using initial estimates of

$$\begin{aligned} \phi &= \phi_2 \\ \lambda &= x/\cos\phi \end{aligned}$$

compute (x_t, y_t) obtained from (x, y) determined by the forward equations. Determine adjusted (ϕ, λ) from:

$$\begin{aligned} \phi &= [(\phi - \phi_1)(y - y_c)/(y_t - y_c)] + \phi_1 \\ \lambda &= \lambda x/x_t \end{aligned}$$

Using new estimates, repeat process until change in each axis reaches tolerance.

6.0.5 Rectangular Polyconic.

If $\phi_{ts} = 0$, then

$$A = \lambda/2$$

otherwise

$$A = \tan[(\lambda \sin\phi)/2] \sin\phi_{ts}.$$

If $\phi = 0$, then

$$\begin{aligned} x &= 2A \\ y &= -\phi_0. \end{aligned}$$

otherwise

$$\begin{aligned} \rho &= \cot\phi \\ \theta &= 2\tan^{-1}(A \sin\phi) \\ x &= \rho \sin\theta \\ y &= \phi - \phi_0 + \rho(1 - \cos\theta). \end{aligned}$$

6.0.6 Modified Polyconic.

For $\phi = 0$

$$\begin{aligned} x &= 2S_n f(\lambda) \\ y &= 0 \end{aligned}$$

otherwise

$$\begin{aligned} x &= \frac{2S_n f(\lambda) V^{S_m/S_n}}{1 + [f(\lambda)\sin\phi V^{S_m/S_n - 1}]^2} \\ y &= S_m \mathrm{M}(\phi) + x f(\lambda) \sin\phi V^{S_m/S_n - 1} \end{aligned}$$

where

$$V = \cos\phi/(1 - e^2 \sin^2\phi)^{1/2}$$

When $\phi_0 = 0$, then

$$f(\lambda) = m_0 \lambda/(2S_n)$$

otherwise

$$f(\lambda) = \frac{\tan[(n_0 \sin\phi_0 \lambda)/(2S_n)]}{\sin\phi_0 V_0^{S_m/S_n - 1}}$$

In the case of $\phi_0 = 0$, typically $S_m = S_n = n_0$ where n_0 is the scale factor along the equator. Where $\phi_0 \neq 0$, K also needs to be specified.

6.0.7 Ginzburg Polyconics.

These polyconics are based upon polynomials in ϕ defining the cartesian coordinates at the central meridian (x_m, y_m) and at the $\lambda = 180°$ meridian (x_l, y_l):

$$\begin{aligned} x_m &= 0 \\ y_m &= \sum_{n=1} c_{2n-1}\phi^{2n-1} \\ x_l &= \sum_{n=0} a_{2n}\phi^{2n} \\ y_l &= \sum_{n=1} b_{2n-1}\phi^{2n-1} \end{aligned}$$

where the coefficents are:

c_1	1.0	1.0	1.0	1.0
c_3	0.045	0.0	0.0	0.0
a_0	$5\pi/6$	2.8284	2.5838	2.6516
a_2	−0.62636	−1.6988	−0.83584	−0.76534
a_4	−0.0344	0.75432	0.17037	0.19123
a_6	0.0	−0.18071	−0.038097	−0.047094
b_1	1.3493	1.76003	1.54331	1.36289
b_3	−0.05524	−0.38914	−0.41143	−0.13965
b_5	0.0	0.042555	0.082742	0.031762

To determine the radius of the polyconic arc, the radius of a circle circumscribing the triangle formed by the $\lambda = \pm 180°$ points and the central merdian:

$$\rho = (x_l^2 + y_s^2)/(2x_l y_s)$$

where $y_s = |y_l - y_m|$ and where the sign of ρ is taken from ϕ. The cartesian coordinates are determined by:

$$\begin{aligned} x &= \rho\sin\theta\lambda \\ y &= \rho\cos\theta\lambda + y_m \end{aligned}$$

where $\theta = \tan^{-1}(x/(\rho - y))/\pi$.

6.0.8 Křovák Oblique Confomal Conic Projection

Projection of geographic coordinates by the oblique conformal conic projection is three step process. First, conversion of the ellipsoid coordinates (ϕ,λ) to coordinates on a conformal sphere (ϕ_c,λ_c) which is followed by translation of the spherical coordinates (ϕ',λ'). Finally, these coordinates are projected to planar coodinates with the tangential form of the conformal conic projection. The following equations are presented in their most general form and include options that may be disregarded in the final application.

Forward projection

In the following conversion from ellipsoid to conformal sphere coodinates the values of the projection origin on the ellipsoid, ϕ_0-λ_0, must be provided. R_c is the radius of the sphere computed as the geometric mean of the meridinal and parallel ellipsoid radii.

$$\begin{aligned}
\phi_c &= 2\arctan\left[K\tan^C(\pi/4+\phi/2)\left(\frac{1-e\sin\phi}{1+e\sin\phi}\right)^{Ce/2}\right]-\pi/2 && (6.18)\\
\lambda_c &= C(\lambda-\lambda_0) && (6.19)\\
R_c &= a\frac{\sqrt{1-e^2}}{1-e^2\sin^2\phi_0} && (6.20)\\
C &= \sqrt{1+\frac{e^2cos^4\phi_0}{1-e^2}} && (6.21)\\
\chi &= \arcsin\left(\frac{\sin\phi_0}{C}\right) && (6.22)\\
K &= \tan(\chi/2+\pi/4)/\left[\tan^C(\phi_0/2+\pi/4)\left(\frac{1-e\sin\phi_0}{1+e\sin\phi_0}\right)^{Ce/2}\right] && (6.23)
\end{aligned}$$

The following is general spherical translation but in this application only the shift of the latitude of the pole, α, is used. Angles β and λ_{c0} are ignored (set to 0).

$$\begin{aligned}
\phi' &= \arcsin(\sin\alpha\sin\phi_c-\cos\alpha\cos\phi_c\cos(\lambda_c-\lambda_{c0})) && (6.24)\\
\lambda' &= \arcsin(\cos\phi_c\sin(\lambda_c-\lambda_{c0})/\cos\phi')+\beta && (6.25)\\
\alpha &= \pi/2-\phi_t && (6.26)
\end{aligned}$$

where ϕ_t is the latitude of the new sphere on the old sphere.

The translated spherical coordinates are now projected to the tangent cone by the general spherical conformal conic projection:

$$\begin{aligned}
n &= \sin\phi_1 && (6.27)\\
F &= \cos\phi_1\tan^n(\pi/4+\phi_1/2)/n && (6.28)\\
\rho &= k_0R_cF/\tan^n(\pi/4+\phi'/2) && (6.29)\\
\rho_0 &= k_0R_cF/\tan^n(\pi/4+\phi_1/2) && (6.30)\\
\theta &= n(\lambda'-\lambda'_0) && (6.31)\\
x &= \rho\sin\theta && (6.32)\\
y &= \rho_0-\rho\cos\theta && (6.33)
\end{aligned}$$

Inverse projection

For the inverse case, coordinates on the conformal sphere are first found from:

$$\phi' = 2\arctan\left(\frac{k_0 R_c F}{\rho}\right)^{1/n} - \pi/2 \tag{6.34}$$

$$\lambda' = \theta/n + \lambda_{c0} \tag{6.35}$$

$$\rho = \pm\sqrt{x^2 + (\rho_0 - y)^2}, \text{ taking sign of } n \tag{6.36}$$

$$\theta = \arctan\left(\frac{x}{\rho_0 - y}\right) \tag{6.37}$$

To revert these coordinates to the unshifted spherical coordinates:

$$\phi_c = \arcsin(\sin\alpha\sin\phi' + \cos\alpha\cos\phi'\cos(\lambda' - \beta)) \tag{6.38}$$

$$\lambda_c = \arcsin(\cos\phi'\sin(\lambda' - \beta)/\cos\phi_c) + \lambda_{c0} \tag{6.39}$$

At this point the ellipsoid coordinates are obtained from

$$\lambda = \lambda_c/C + \lambda_0 \tag{6.40}$$

$$\phi_i = 2\arctan\left[\frac{\tan^{1/C}(\phi'/2 + \pi/4)}{K^{1/C}\left(\frac{1 - e\sin\phi_{i-1}}{1 + e\sin\phi_{i-1}}\right)^{e/2}}\right] - \pi/2 \tag{6.41}$$

with the initial value of $\phi_{i-1} = \phi_c$ and ϕ_{i-1} iteratively replaced by ϕ_i until $|\phi_i - \phi_{i-1}|$ is less than an acceptable error value.

The Křovák Projection Grid

The following script defines the execution of the program *proj* to compute coordinates for the *S-JTSK* grid system covering the states of the Czech Republic and Slovak Republic. The central (or origin) longitude is specified as being $42°30'$ east of the a point off the isle of Ferro (Hierro) in the Canary Islands. The Ferro point is at $17°39'59.7354''$W but the value is often rounded to $17°40'W$ for topographic work. The latitude of origin on the ellipsoid is $49°30'$. The latitude of the translated pole on the original conformal sphere is $56°42'42.69689''$ and the latitude of the cone's point of tangency on the translated sphere is $78°30'$.

```
#<S-JTSK> Krovak Coordinate System
proj +proj=kocc +ellps=bessel +czech
     +lon_0=42d30
     +lat_0=49d30
     +lat_t=59d42'42.69689
     +lat_1=78d30
     +k_0=.9999
```

The natural math conversion puts the coordinates of the area in the $-x, -y$ quadrant. The S-JTSK projection, however, uses positive y to the left of the longitude of origin and positive x south of the cone's polar point near Helsinki. The option `+czech` converts to S-JTSK x, y output.

6.0.9 Lambert Conformal Conic Alternative Projection

`+proj=lcca` This tangential conic projection is a variant of the Lambert Conformal Conic that was employed by the French and several north African and near eastern countries. The unique problem was that the projection was computed with a severely truncated series which compromised its conformality as well as creating confusion.

Forward projection

The following are the equations to determine the planar coordinates from geographic coordinates.

$$
\begin{aligned}
x &= r \sin\theta && (6.42)\\
y &= r_0 - r\cos\theta && (6.43)\\
\theta &= l\lambda && (6.44)\\
l &= \sin\phi_0 && (6.45)\\
r &= r_0 \mp \Delta r && (6.46)\\
\Delta r &= F(S) = S + \frac{S^3}{6R_0N_0}\left\{\pm\frac{S^4(5R_0-4N_0)\tan\phi_0}{24R_0^2N_0^2}+\right. \\
&\left. \frac{S^5(5+3\tan^2\phi_0)}{120R_0N_0^3} \pm \frac{S^6(7+4\tan^2\phi_0)\tan\phi_0}{240R_0N_0^4}\right\} && (6.47)\\
S &= M(\phi) - M(\phi_0) && (6.48)\\
r_0 &= N_0/\tan\phi_0 && (6.49)\\
N_0 &= a/\sqrt{1-e^2\sin^2\phi_0} && (6.50)\\
R_0 &= \frac{a(1-e^2)}{(1-e^2\sin^2\phi_0)^{3/2}} && (6.51)
\end{aligned}
$$

where $M(\phi)$ is the meridinal arc distance from the equator to latitude ϕ.

In the case of this "nearly conformal" projection, only the first two terms of (6.47) are evaluated. Even the remaining series coefficients (inside curly braces) are an approximation where the higher order terms were simplified by assuming $R_0 = N_0$.

Inverse projection

The geographic coordinates are obtained from the planar cartesian by:

$$
\begin{aligned}
\theta &= \arctan\left(\frac{x}{r_0 - y}\right) && (6.52)\\
\Delta r &= y - x\tan\left(\frac{\theta}{2}\right) && (6.53)\\
\lambda &= \theta/\sin\phi_0 && (6.54)
\end{aligned}
$$

The value of S can be obtained by applying Newton-Raphson's method to (6.47):

$$S_{i+1} = S_i - \frac{F(S_i) - \Delta r}{F'(S_i)} \tag{6.55}$$

where the initial value of $S_i = \Delta r$ and iteration is continued until specified tolerance is met. Finally, latitude is obtained from the inverse meridinal arc routine:

$$\phi = M^{-1}(S + M(\phi_0)); \tag{6.56}$$

PROJ.4 usage

Projection selection is `+proj=lcca` where only `+lat_0=` is used to specify point of cone tangency and mathematical origin (along with `+lon_0`). For a secant cone, use the scale factor option `+k_0=`.

For an accurate, complete Lambert Conformal Conic use `+proj=lcc`.

6.0.10 Hall Eucyclic.

`+proj=hall [+K=] or [+beta=]` Fig. 6.1 Ref. [16]
Without specifying options K assumes the value of 1. For computing the Maurer SNo. 73 projection, set `beta=45d`. In all cases compute

$$\sin\beta = \frac{1}{1+K} \tag{6.57}$$

$$A = 2\sqrt{\frac{\pi}{\pi + 4\beta(1+K)}} \tag{6.58}$$

$$\rho_0 = \frac{A}{2}\left(1 + K + \sqrt{K(2+K)}\right) \tag{6.59}$$

When $|\phi| \neq \pi/2$ use Newton-Raphson iteration to determine θ from

$$\begin{aligned} P(\theta) &= \theta - K^2\beta - (1+K)\sin\theta \\ &\quad + \left(1 + (1+K)^2 - 2(1+K)\cos\theta\right)\left(\beta + \arctan\frac{\sin\theta}{1+K-\cos\theta}\right) \\ &\quad - \frac{1}{2}(1-\sin\phi)[\pi + 4\beta(1+K)] \end{aligned} \tag{6.60}$$

$$P'(\theta) = 2(1+K)\sin\theta\left(\beta + \arctan\frac{\sin\theta}{1+K-\cos\theta}\right) \tag{6.61}$$

and then

$$\rho = A\sqrt{1 + (1+K)^2 - 2(1+K)\cos\theta} \tag{6.62}$$

$$\beta_1 = \arctan\frac{\sin\theta}{1+K-\cos\theta} \tag{6.63}$$

Otherwise $\beta_1 = 0$ and

$$\rho = \begin{cases} AK & \text{if } \phi = \pi/2 \\ A(K+2) & \text{if } \phi = -\pi/2 \end{cases} \tag{6.64}$$

Finally

$$\omega = \lambda\frac{(\beta_1 + \beta)}{\pi} \tag{6.65}$$

$$x = \rho\sin\omega \tag{6.66}$$

$$y = \rho_0 - \rho\cos\omega \tag{6.67}$$

where ρ_0 is determined from ϕ_0.

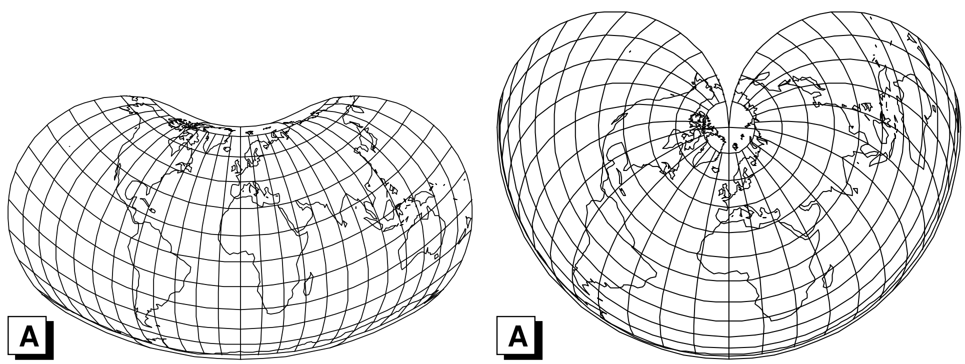

Figure 6.1: A–Hall Eucyclic and B–Maurer SNo. 73 (`+proj=hall +K=0`)

Chapter 7

Azimuthal Projections

7.1 Perspective

7.1.1 Perspective Azimuthal Projections.

The term perspective is applied to several of the azimuthal projections as well as a few conic and cylindrical projections. Figure 7.1 shows the geometry of the perspective projection where a ray originating at the "perspective" point **P** passes though the object to be plotted at **L** to the plane of the map at **L**′ at a distance ρ from the point of tangency of the plane with the sphere. From the known conditions, ψ and h the the distance ρ is determined by the general expression

$$\rho = \frac{(1+h)\sin\psi}{h+\cos\psi} \tag{7.1}$$

From this equation three of the common perspective azimuthal projections are simplified special cases of h:

$$\rho = \begin{cases} \tan\psi & h = 0 \text{ Gnomonic} \\ \dfrac{2\sin\psi}{1+\cos\psi} = 2\tan(\psi/2) & h = 1 \text{ Stereographic} \\ \sin\psi & h = \infty \text{ Orthographic} \end{cases} \tag{7.2}$$

The angle ψ' is the limit of ψ for the visible part of the projection and is defined by:

$$\psi' = \begin{cases} \arccos(-1/h) & |h| > 1 \\ \pi/2 + \arcsin h & |h| \le 1 \end{cases} \tag{7.3}$$

For the case where $|h| \le 1$ then $\rho = \infty$ when $\psi = \psi'$.

For the case where **TS** is the polar axis the angle ψ becomes the colatitude and equation **??** becomes

$$\rho = \begin{cases} \cot\phi & \text{Gnomonic} \\ 2\tan(\pi/4 - \phi/2) & \text{Sterographic} \\ \cos\phi & \text{Orthographic} \end{cases} \tag{7.4}$$

The Cartesian coordinates polar aspect

$$x = \rho\sin\lambda \qquad y = \mp\rho\cos\lambda \tag{7.5}$$

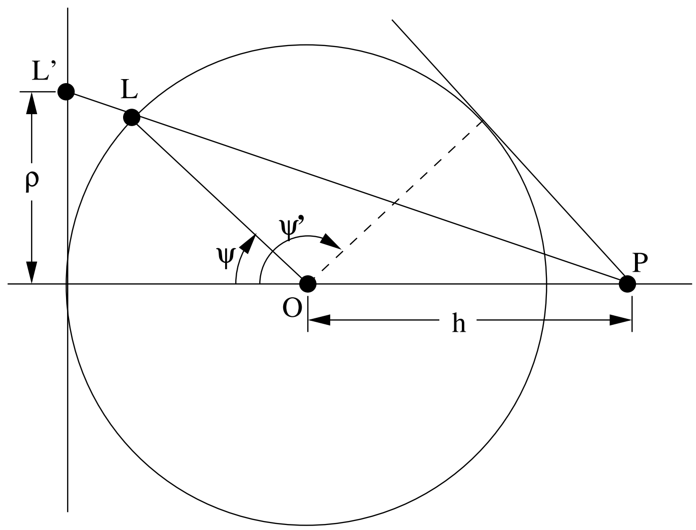

Figure 7.1: Geometry of perspective projections

where $\lambda = 0$ follows the negative y azis for the North polar aspect and the positive y axis for the South polar aspect.

For the oblique aspect ψ or the angular distance of **L** from the center of the projection $(\phi_0, \lambda = 0)$ to the projected point is the geodesic or "Great Circle" distance. Snyder in both [14] and [17] has used:

$$\cos\psi = \sin\phi_0 \sin\phi + \cos\phi_0 \cos\phi \cos\lambda \tag{7.6}$$

However, for better precision near the origin ([14, p. 30]):

$$\cos(\psi/2) = \left[\sin^2\left(\frac{\phi - \phi_0}{2}\right) + \cos\phi_0 \cos\phi \sin^2\frac{\lambda}{2}\right]^{1/2} \tag{7.7}$$

The azimuth from the projection center to the point is:

$$\sin\alpha = \sin\lambda\cos\phi / \sin\psi \text{ or} \tag{7.8}$$
$$\cos\alpha = (\cos\phi_1 \sin\phi - \sin\phi_1 \cos\phi\cos\lambda)/\sin\psi \tag{7.9}$$

The oblique coordinates x, y are

$$x = K\cos\phi\sin\lambda \tag{7.10}$$
$$y = K(\cos\phi_1 \sin\phi - \sin\phi_1 \cos\phi\cos\lambda) \tag{7.11}$$

where

$$K = \begin{cases} 1/(\sin\phi_1 \sin\phi + \cos\phi_1 \cos\phi\cos\lambda) & \text{Gnomonic} \\ 2/(1 + \sin\phi_1 \sin\phi + \cos\phi_1 \cos\phi\cos\lambda) & \text{Stereographic} \\ 1 & \text{Orthographic} \end{cases} \tag{7.12}$$

Further simplifications for the equitorial case are obvious.

For the inverse projection the first operation is to determine

$$\rho = \sqrt{x^2 + y_2} \tag{7.13}$$

If $\rho = 0$ then $\lambda = 0$ and $\phi = \phi_0$. Otherwise

$$\psi = \begin{cases} \arctan \rho & \text{Gnomonic} \\ 2 \arctan \dfrac{\rho}{2k_0} & \text{Stereographic} \\ \arcsin \rho & \text{Orthographic} \end{cases} \tag{7.14}$$

Geographic coordinates are now obtained from

$$\phi = \arcsin \left[\cos \psi \sin \phi_0 + (y \sin \psi \cos \phi_0)/\rho\right] \tag{7.15}$$
$$\lambda = \operatorname{atan2}(x \sin \psi, \rho \cos \phi_0 \cos \psi - y \sin \phi_0 \sin \psi) \tag{7.16}$$

If $|\phi_0| = 90°$ then

$$\lambda = \operatorname{atan2}(x, \mp y) \tag{7.17}$$

where y takes the opposite sign of ϕ_0.

7.1.2 Stereographic Projection.

```
+proj=stere
+proj=sterea
+proj=ups
+proj=rouss
```

The conformal Stereographic projection is useful for both mapping of continental size regions as well as grid systems with a near circular perimeter. Although spherical form, useful for small scale projections has only one set of equations, three different forms of the ellipsoid Oblique Stereographic projection are available. Two of them are based upon conformal coversion of the geographic coordinates on the ellipsoid to coodinates on the sphere while the third uses a polynomial approximation. For the polar aspect, only one ellipsoidal method is used in the `+proj=stere` version and the specialized use of the polar projection in the Universal Polar Stereographic system is available with `+proj=ups`.

Spherical Stereographic

The forward spherical oblique equations ($0 \leq |\phi_0| \leq \pi/2$) are:

$$x = 2k \cos \phi \sin \lambda \tag{7.18}$$
$$y = 2k(\cos \phi_0 \sin \phi - \sin \phi_0 \cos \phi \cos \lambda) \tag{7.19}$$

where

$$k = k_0/(1 + \sin \phi_0 \sin \phi + \cos \phi_0 \cos \phi \cos \lambda) \tag{7.20}$$

For the equitorial aspect, $\phi_0 = 0$,

$$y = k \sin \phi \tag{7.21}$$
$$k = 2k_0/(1 + \cos \phi \cos \lambda) \tag{7.22}$$

and where x is obtained from equation 7.18.

For the polar aspect, $\phi_0 \pm \pi/2$, the equations simplify to:

$$t = \tan \left(\frac{\pi}{4} - f\frac{\phi}{2}\right) \tag{7.23}$$
$$x = 2k_0 t \sin \lambda \tag{7.24}$$
$$y = 2fk_0 t \cos \lambda \tag{7.25}$$

where $f \mp 1$ is assigned the opposite sign of ϕ_0.

To determine the inverse spherical projection determine:

$$\rho = (x^2 + y^2)^{1/2} \tag{7.26}$$

$$c = 2\arctan\left(\frac{\rho}{2k_0}\right) \tag{7.27}$$

If $\rho = 0$ then $\phi = \phi_0$ and $\lambda = 0$ otherwise for the general oblique case:

$$\phi = \arcsin\left(\cos c \sin\phi_0 + \frac{y \sin c \cos\phi_0}{\rho}\right) \tag{7.28}$$

$$\lambda = \arctan 2\left(\frac{x \sin c}{\rho\cos\phi_0 \cos c - y \sin\phi_0 \sin c}\right) \tag{7.29}$$

or for the polar case

$$\lambda = \arctan 2\left(\frac{x}{fy}\right) \tag{7.30}$$

where $f \pm 1$ with the sign of ϕ_0. For the equitorial case:

$$\lambda = \arctan 2\left(\frac{x \sin c}{\rho\cos\phi_0 \cos c}\right) \tag{7.31}$$

In the case of `+proj=stere` the specification of the latitude origin (`+lat_0=`ϕ_0) determines the oblique or polar mode of usage. Scaling may be performed by using `k_0=`k_0 in all cases or latitude of true scale (`+lat_ts=`ϕ_{ts}) for the polar case. Note that the central meridian for the southern polar case runs from the projection origin to the north.

The Universal Polar Stereographic is much like the Universal Transverse Mercator system where scaling and false easting/northings are all predefined and an ellipsoid must be specified.

Oblique Stereographic using intermediate sphere.

Using the spherical stereographic projection:

$$x = 2kR_c \cos\chi \sin(\lambda_c) \tag{7.32}$$

$$y = 2kR_c[\cos\chi_0 \sin\chi - \sin\chi_0 \cos\chi \cos(\lambda_c)] \tag{7.33}$$

where

$$k = k_0/[1 + \sin\chi_0 \sin\chi + \cos\chi_0 \cos\chi \cos(\lambda_c)] \tag{7.34}$$

The difference between `stere` and `sterea` is how the conformal latitudes χ and χ_0 and longitude λ_c and radius R_c are determined.

For determining the χ, χ_0 and R_c values the function series `pj_sgauss` and `pj_gauss` are used for the respective `stere` and `sterea` entries (see section 3.3).

For the inverse case:

$$\rho = (x^2 + y^2)^{1/2} \tag{7.35}$$

$$c = 2\arctan\left(\frac{\rho}{2R_c k)0}\right) \tag{7.36}$$

$$\chi = \left(\cos c \sin\chi_0 + \frac{y \sin c \cos\chi_0}{\rho}\right) \tag{7.37}$$

$$\lambda = \arctan\left(\frac{x \sin c}{\rho\cos\chi_0 \cos c - y \sin\chi_0 \sin c}\right) \tag{7.38}$$

Where $\rho = 0$ then $\chi = \chi_)$ and $\lambda_c = 0$. For the geographic coordinates execute the inverse confomal functions `pj_sgauss_inv` or `pj_gauss_inv`.

Oblique Roussilhe Stereographic.

Another oblique version of the stereographic projection for the ellipsoid presented by Roussilhe [12]. Given:

$$\begin{aligned} s &= \int_{\phi_0}^{\phi} M d\phi && (7.39) \\ \alpha &= \lambda N \cos\phi && (7.40) \end{aligned}$$

where M and N are the respective ellipsoid meridional radius and radius normal to the meridian:

$$\begin{aligned} M &= (1-e^2)(1-e^2\sin^2\phi)^{-3/2} \\ N &= (1-e^2\sin^2\phi)^{-1/2} \end{aligned}$$

then the Cartesian coordinates are computed by:

$$\begin{aligned} x &= \alpha + A_1\alpha s^2 - A_2\alpha^3 - A_3\alpha^3 s + A_4\alpha s^4 - A_5\alpha^3 s^2 - A_6\alpha^5 && (7.41) \\ y &= s + B_1\alpha^2 + B_2 s^3 + B_3\alpha^2 s + B_4\alpha^4 + B_5\alpha^2 s^2 - B_6\alpha^4 s + \\ &\quad B_7\alpha^2 s^3 + B_8 s^5 && (7.42) \end{aligned}$$

and where:

$$t_0 = \tan\phi_0$$

$$\begin{aligned} A_1 &= \frac{1}{4M_0N_0} & B_1 &= \frac{t_0}{2N_0} \\ A_2 &= \frac{2t_0^2 - 1 - 2e^2\sin^2\phi_0}{12M_0N_0} & B_2 &= \frac{1}{12M_0N_0} \\ A_3 &= \frac{t_0(1+4t_0^2)}{12M_0N_0^2} & B_3 &= \frac{1+2t_0^2-2e^2\sin^2\phi}{4M_0N_0} \\ A_4 &= \frac{1}{24M_0^2N_0^2} & B_4 &= \frac{t_0(2-t_0^2)}{24M_0N_0^2} \\ A_5 &= \frac{12t_0^4+11t_0^2-1}{24M_0^2N_0^2} & B_5 &= \frac{t_0(5+4t_0^2)}{8M_0N_0^2} \\ A_6 &= \frac{-2t_0^4+11t_0^2-2}{240M_0^2N_0^2} & B_6 &= \frac{6t_0^4-5t_0^2-2}{48M_0^2N_0^2} \\ & & B_7 &= \frac{12t_0^4+19t_0^2+5}{24M_0^2N_0^2} \\ & & B_8 &= \frac{1}{120M_0^2N_0^2} \end{aligned}$$

The distance s is obtained from the meridional distance function `pj_mdist` (3.2) by initializing s_0 with the meridional distance at ϕ_0 and subtracting it from the meridional distance for each value of ϕ.

For the inverse projection first determine:

$$\begin{aligned} \alpha &= x - C_1xy^2 + C_2x^3 + C_3x^3y - C_4x^5 + C_5x^3y^2 + C_6xy^4 - \\ &\quad C_7x^5y - C_8x^3y^3 && (7.43) \\ s &= y - D_1x^2 - D_2Y^3 - D_3x^xy + D_4x^4 - D_5x^2y^2 + D_6x^4y - D_7x^2y^3 + \\ &\quad D_8y^5 - D_9x^6 + D_{10}x^4y^2 + D_{11}x^2y^4 && (7.44) \end{aligned}$$

where

$$
\begin{aligned}
C_1 &= \frac{1}{4M_0N_0} & D_1 &= \frac{t_0}{2N_0} \\
C_2 &= \frac{2t_0^2 - 1 - 2e^2\sin^2\phi_0}{12M_0N_0} & D_2 &= \frac{1}{12M_0N_0} \\
C_3 &= \frac{t_0(1+t_0^2)}{3M_0N_0^2} & D_3 &= \frac{1+2t_0^2-2e^2\sin^2\phi_0}{4M_0N_0} \\
C_4 &= \frac{22t_0^4+34t_0^2-3}{240M_0^2N_0^2} & D_4 &= \frac{t_0(1+t_0^2)}{8M_0N_0^2} \\
C_5 &= \frac{12t_0^4+13t_0^2+4}{24M_0^2N_0^2} & D_5 &= \frac{t_0(1+2t_0^2)}{4M_0N_0^2} \\
C_6 &= \frac{1}{16M_0^2N_0^2} & D_6 &= \frac{6t_0^4+6t_0^2+1}{16M_0^2N_0^2} \\
C_7 &= \frac{t_0(16t_0^4+33t_0^2+11)}{48M_0^2N_0^3} & D_7 &= \frac{t_0^2(3+4t_0^2)}{8M_0^2N_0^2} \\
C_8 &= \frac{t_0(4t_0^2+1)}{36M_0^2N_0^3} & D_8 &= \frac{1}{80M_0^2N_0^2} \\
& & D_9 &= \frac{t_0(-26t_0^4+178t_0^2-21)}{720M_0^2N_0^2} \\
& & D_{10} &= \frac{t_0(48t_0^4+86t_0^{+}29)}{96M_0^2N_0^3} \\
& & D_{11} &= \frac{t_0(44t_0^2+37)}{96M_0^2N_0^3}
\end{aligned}
$$

Determine the latitude from the inverse meridional function `pj_inv_mdist` for $s+s_0$ and determine longitude from $\lambda = \alpha(1-e^2\sin^2\phi)^{1/2}/\cos\phi$

7.2 Modified

7.2.1 Hammer and Eckert-Greifendorff.

`+proj=hammer [+W=]` Fig. ??

$$W = \begin{cases} 0.5 & \text{Hammer (\texttt{+W} when not given)} \\ 0.25 & \text{\texttt{W=0.25} for Eckert-Greifendorff (fig. ??)} \end{cases} \tag{7.45}$$

$$D = \left(\frac{2}{1+\cos\phi\cos(W\lambda)}\right)^{\frac{1}{2}} \tag{7.46}$$

$$x = \frac{D}{W}\cos\phi\sin(W\lambda) \tag{7.47}$$

$$y = D\sin\phi \tag{7.48}$$

7.2.2 Aitoff, Winkel Tripel and with Bartholomew option.

`+proj=aitoff`
`+proj=wintri [lat_1=]`
The formulas for Aitoff:

$$\delta = \arccos\left(\cos\phi\cos\frac{\lambda}{2}\right) \tag{7.49}$$

If $\delta = 0$, then $x = y = 0$ else

$$\cos\alpha = \frac{\sin\phi}{\sin\delta} \tag{7.50}$$
$$x = \pm\delta\sin\alpha \tag{7.51}$$
$$y = \delta\cos\alpha \tag{7.52}$$

where x takes the sign of λ.

For Winkel Tripel the values for (x_1, y_1) are determined from above and the values (x_2, y_2) are determined from the Equirectangular projection repeated here:

$$\phi_1 = \begin{cases} 50.467^\circ & \text{Winkel Tripel (\texttt{+lat\_1=} not defined)} \\ 40^\circ & \text{\texttt{+lat\_1=40} for Bartholomew (fig. 7.2)} \end{cases} \tag{7.53}$$
$$x_2 = \lambda\cos\phi_1 \tag{7.54}$$
$$y_2 = \phi \tag{7.55}$$

Resultant value is:

$$x = \frac{x_1 + x_2}{2} \tag{7.56}$$
$$y = \frac{y_1 + y_2}{2} \tag{7.57}$$

7.2.3 Wagner VII (Hammer-Wagner) and Wagner VIII.

Name	+proj=	Fig.	Ref.
Wagner VII	wag7	??	
Wagner VIII	wag8	??	

For $n = 1/3$, initialization for Wagner VII:

$$m_2 = 1 \tag{7.58}$$
$$m_1 = \sin 65^\circ \tag{7.59}$$
$$k = 2\sqrt{\sin 32.5^\circ} \tag{7.60}$$
$$C_x = \frac{2k}{\sqrt{nm_1}} \tag{7.61}$$
$$C_y = \frac{2}{k\sqrt{nm_1}} \tag{7.62}$$

and for Wagner VIII:

$$m_2 = \frac{\arccos(1.2\cos 60^\circ)}{60^\circ} \tag{7.63}$$
$$m_1 = \frac{sin 65^\circ}{\sin(m_2 90^\circ)} \tag{7.64}$$
$$k = \sqrt{\frac{2\sin 32.5^\circ}{\sin 30^\circ}} \tag{7.65}$$
$$C_x = \frac{2k}{\sqrt{m_1 m_2 n}} \tag{7.66}$$
$$C_y = \frac{2}{k\sqrt{m_1 m_2 n}} \tag{7.67}$$

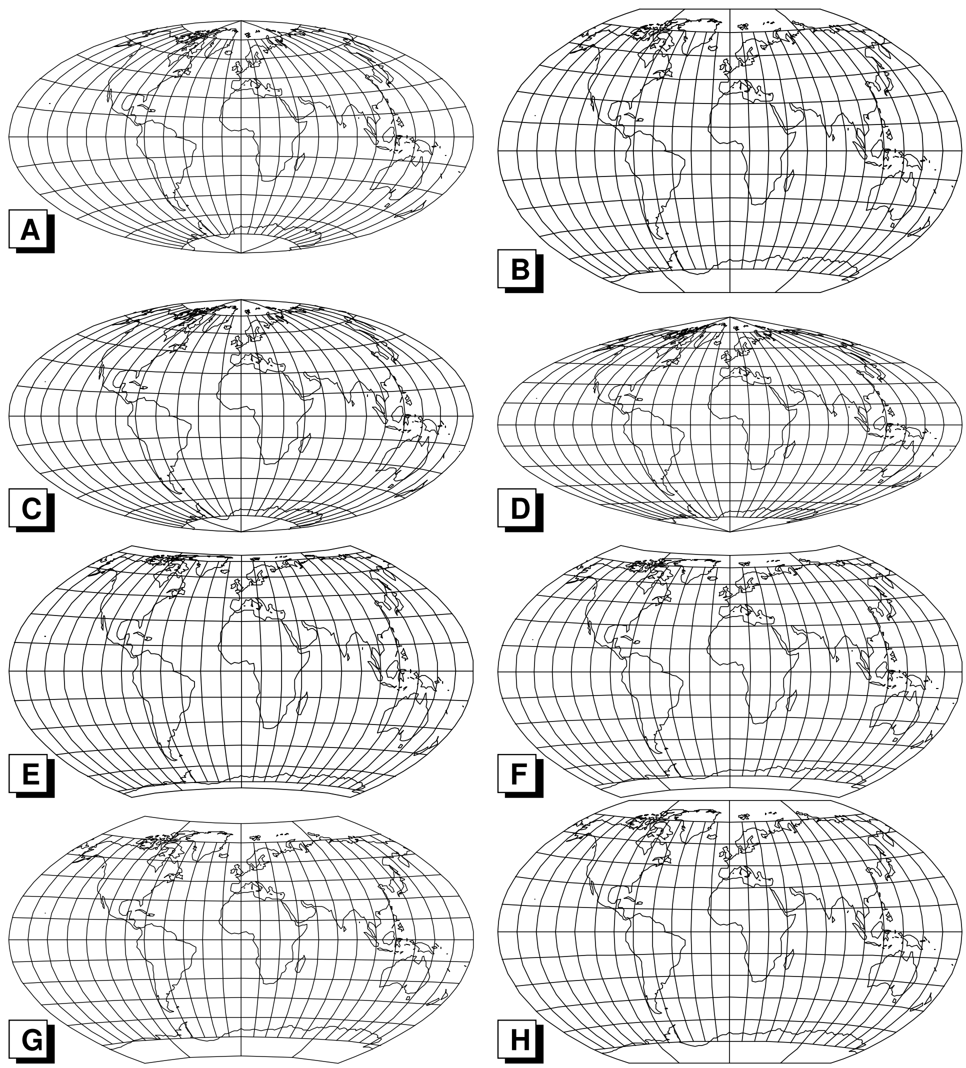

Figure 7.2: Modified Azimuthals.
A–Aitoff, **B**–Winkel Tripel, **C**–Hammer, **D**–Eckert-Greifendorff (`+proj=hammer +W=0.25`), **E**–Wagner VII, **F**–Wagner VIII, **G**–Wagner IX and **G**–Bartholomew (`+proj=wintri +lat_1=40`).

Common computations [22, p. 205–207]:

$$\sin\psi = m_1 \sin(m_2\phi) \tag{7.68}$$

$$\cos\delta = \cos\psi\cos\frac{\lambda}{3} \tag{7.69}$$

If $\delta = 0$ then $x = y = 0$ else

$$\cos\alpha = \frac{\sin\psi}{\sin\delta} \tag{7.70}$$

$$x = \pm C_x \sin\frac{\delta}{2}\sin\alpha \tag{7.71}$$

$$y = C_y \sin\frac{\delta}{2}\cos\alpha \tag{7.72}$$

where x assumes the sign of λ. An alternate, more efficient method [17, p. 233] for the common computations are:

$$S = m1 * \sin(m2 * \phi) \tag{7.73}$$
$$C_0 = \sqrt{1 - S^2} \tag{7.74}$$
$$C_1 = \left[\frac{2}{1 + C_0 \cos(\lambda/3)}\right]^{\frac{1}{2}} \tag{7.75}$$
$$x = \frac{C_x}{2} C_0 C_1 \sin(\lambda/3) \tag{7.76}$$
$$y = \frac{C_y}{2} S C_1 \tag{7.77}$$

7.2.4 Wagner IX (Aitoff-Wagner).

`+proj=wag9` Fig. ??

$$n = \frac{5}{18} \tag{7.78}$$
$$m = \frac{7}{9} \tag{7.79}$$
$$k = \sqrt{\frac{14}{5}} \tag{7.80}$$
$$\psi = m\phi \tag{7.81}$$
$$\delta = \arccos[\cos(n\lambda)\cos\psi] \tag{7.82}$$

If $\delta = 0$ then $x = y = 0$ else

$$\cos\alpha = \frac{\sin\psi}{\sin\delta} \tag{7.83}$$
$$x = \pm\frac{k}{\sqrt{mn}} \delta \sin\alpha \tag{7.84}$$
$$y = \frac{1}{k\sqrt{mn}} \delta \cos\alpha \tag{7.85}$$

where x takes the sign of λ.

7.2.5 Gilbert Two World Perspective.

`gilbert [lat_1=]` Fig. ?? Ref: [3]

$$x = \cos\phi' \sin\lambda' \tag{7.86}$$
$$y = \cos\phi'_1 \sin\phi' - \sin\phi'_1 \cos\phi' \cos\lambda' \tag{7.87}$$
$$D = \sin\phi'_1 \sin\phi' + \cos\phi'_1 \cos\phi' \cos\lambda' \tag{7.88}$$

where $D \geq 0$ for points to be visible and

$$\phi' = \arcsin\tan\left(\frac{\phi}{2}\right) \tag{7.89}$$
$$\lambda' = \frac{\lambda}{2} \tag{7.90}$$

The latitude ϕ_1 is the latitude of perspective azimuth for the oblique case which has a default value of 5°.

Chapter 8

Miscellaneous Projections

This category is a collection of projections that often defy the process of classification. Some are termed "Globular" as the meridians at $\pm 90°$ of the central meridian are circular and usually form the boundaries of a hemispherical plot. Others might be considered Pseudocylindricals and variants of conics but by tradition end up being classified as miscellaneous. Some projections seem like simply cartoons.

8.1 Spherical Forms

8.1.1 Apian Globular II (Arago).

`+proj=apian2` Ref. [15][p. 104]
Early (1524) projection also credited to Arago (1835). Equations based upon description of "equidistant elliptical meridians and equidistant straight parallels." This projection is similar to Apian I within the hemisphere—see comparison figure 8.1

$$y = \phi \qquad\qquad x = \frac{2\lambda}{\pi}\sqrt{\left(\frac{\pi}{2}\right)^2 - \phi^2} \tag{8.1}$$

8.1.2 Apian Globular I, Bacon and Ortelius Oval.

Name	`+proj=`	figure	Ref.
Apian Globular I	apian1	8.2	[17][p. 234]
Bacon Globular	bacon	8.2	[17][p. 234]
Ortelius Oval	ortel	8.3	[17][p. 235]

$$y = \begin{cases} \phi & \text{Apian and Ortelius} \\ \frac{\pi}{2}\sin\phi & \text{Bacon} \end{cases} \tag{8.2}$$

$$F = \begin{cases} \left(\dfrac{(\pi/2)^2}{|\lambda|} + \lambda\right)/2 & \\ \pi/2 & \text{Ortelius when } |\lambda| > \pi/2 \end{cases} \tag{8.3}$$

$$x = \begin{cases} 0 & \text{if } \lambda = 0 \\ \pm\left(|\lambda| - F + (F^2 - y^2)^{\frac{1}{2}}\right) & \text{if } \lambda \neq 0) \end{cases} \tag{8.4}$$

where x takes the sign of λ.

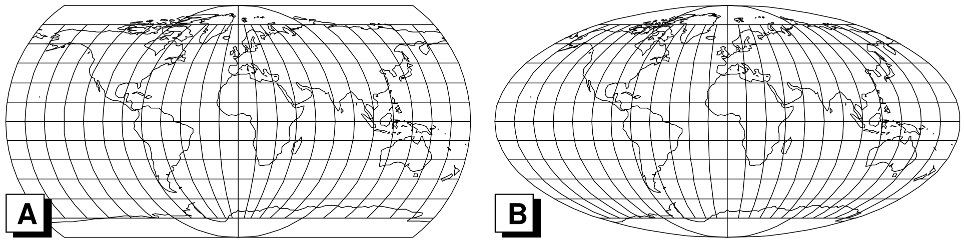

Figure 8.1: Apian Comparison
Global plot of **A**–Apian I and **B**–Apian II.,

8.1.3 Armadillo.

`+proj=arma` Fig. 8.3 Ref. [17, p. 238]
First determine

$$\phi_s = -\arctan\left(\frac{\cos(\lambda/2)}{\tan 20^\circ}\right) \tag{8.5}$$

then if $\phi \geq \phi_s$ then

$$x = (1 + \cos\phi)\sin\frac{\lambda}{2} \tag{8.6}$$

$$\begin{aligned} y = {} & (1 + \sin 20^\circ - \cos 20^\circ)/2 + \sin\phi\cos 20^\circ \\ & - (1 + \cos\phi)\sin 20^\circ\cos(\lambda/2) \end{aligned} \tag{8.7}$$

else point invisible.

8.1.4 August Epicycloidal.

`+proj=august` Fig. 8.3 Ref. [17, p. 235]

$$C_1 = \left(1 - \tan^2\frac{\phi}{2}\right)^{\frac{1}{2}} \tag{8.8}$$

$$C = 1 + C_1\cos\frac{\lambda}{2} \tag{8.9}$$

$$x_1 = \frac{C_1}{C}\sin\frac{\lambda}{2} \tag{8.10}$$

$$y_1 = \frac{\tan(\phi/2)}{C} \tag{8.11}$$

$$x = \frac{4}{3}x_1(3 + x_1^2 - 3y_1^2) \tag{8.12}$$

$$y = \frac{4}{3}y_1(3 + 3x_1^2 - y_1^2) \tag{8.13}$$

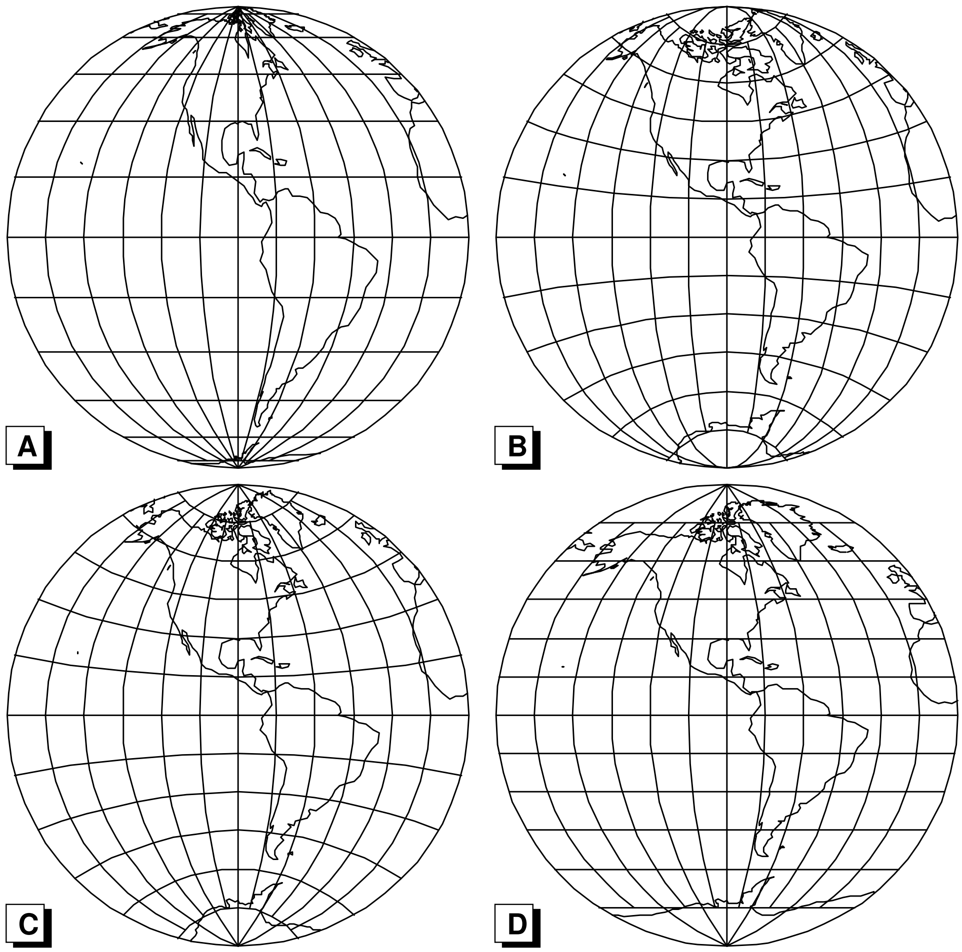

Figure 8.2: Globular Series
A–Bacon Globular, **B**–Fournier Globular 1, **C**–Nicolosi Globular and **D**–Apian Globular I.

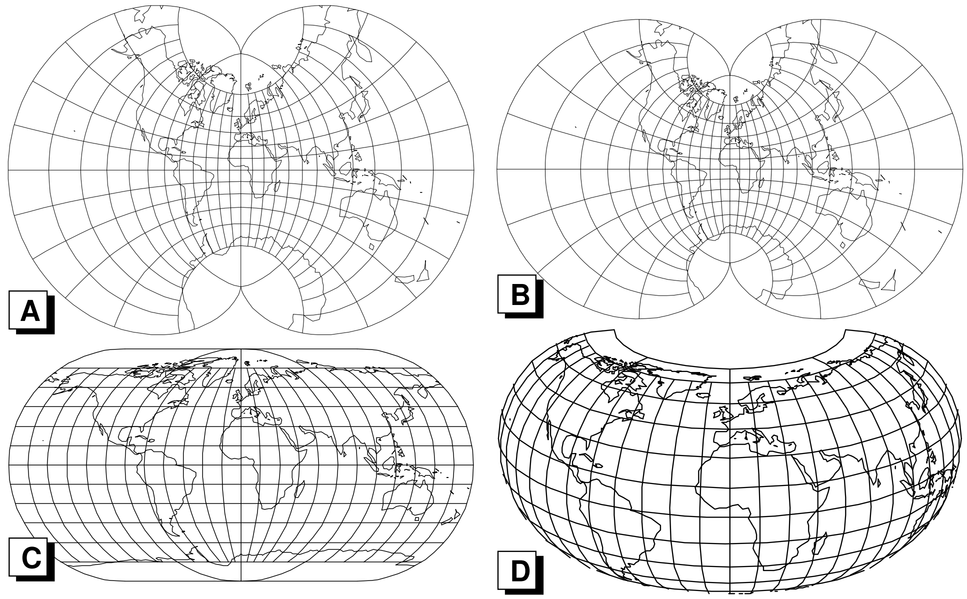

Figure 8.3: General Miscellaneous
A–August Epicycloidal, **B**–Eisenlohr, **C**–Ortelius Oval and **D**–Armadillo.

8.1.5 Eisenlohr

`+proj=eisen` Fig. 8.3 Ref. [17, p. 235]

$$S_1 = \sin\frac{\lambda}{2} \tag{8.14}$$

$$C_1 = \cos\frac{\lambda}{2} \tag{8.15}$$

$$Q = \cos\frac{\phi}{2} \tag{8.16}$$

$$T = \frac{\sin(\phi/2)}{Q + (2\cos\phi)^{\frac{1}{2}} C_1} \tag{8.17}$$

$$C = \left(\frac{2}{1+T^2}\right)^{\frac{1}{2}} \tag{8.18}$$

$$P = \sqrt{\frac{\cos\phi}{2}} \tag{8.19}$$

$$V = \left[\frac{Q + P(C_1 + S_1)}{Q) + P(C_1 - S_1)}\right]^{\frac{1}{2}} \tag{8.20}$$

$$x = (3 + 8^{\frac{1}{2}})(-2\ln V + C(V - 1/V)) \tag{8.21}$$

$$y = (3 + 8^{\frac{1}{2}})(-2\arctan T + CT(V + 1/V)) \tag{8.22}$$

8.1.6 Fournier Globular I.

`+proj=four1` Fig. 8.2 Ref. [17, p. 234]
If $\lambda = 0$ or $|\phi| = \pi/2$ then

$$x = 0 \qquad y = \phi \tag{8.23}$$

else if $\phi = 0$ then

$$x = \lambda \qquad y = 0 \tag{8.24}$$

else if $|\lambda| = \pi/2$ then

$$x = \lambda \cos\phi \qquad y = \frac{\pi}{2}\sin\phi \tag{8.25}$$

otherwise

$$C = \frac{\pi^2}{4} \tag{8.26}$$

$$P = |\pi \sin\phi| \tag{8.27}$$

$$S = \frac{C - \phi^2}{P - 2|\phi|} \tag{8.28}$$

$$A = \frac{\lambda^2}{C} - 1 \tag{8.29}$$

$$y = \pm\left(\{S^2 - A(C - PS - \lambda^2)\}^{\frac{1}{2}} - S\right)/A \tag{8.30}$$

$$x = \pm\lambda\sqrt{1 - \frac{y^2}{C}} \tag{8.31}$$

where x and y take the respective signs of λ and ϕ.

8.1.7 Guyou and Adams Series

Name	+proj=	Figure	Ref.
Guyou	`guyou`	8.4	[17, p. 235–236]
Adams Hemisphere in a Square	`adams_hemi`		
Adams World in a Square I	`adams_wsI`		
Adams World in a Square II	`adams_wsII`		

Several projections have common usage of the elliptical integral of the first kind and are collected under this section.

For the **Guyou** projection: If $|\phi| = \pi/2$, then

$$x = 0 \tag{8.32}$$

$$y = \pm 1.85407 \quad \text{taking the sign of } \phi \tag{8.33}$$

else where $|\lambda| \le \pi/2$

$$\cos a = (\cos\phi\sin\lambda - \sin\phi)/\sqrt{2} \tag{8.34}$$

$$\cos b = (\cos\phi\sin\lambda + \sin\phi)/\sqrt{2} \tag{8.35}$$

$$S_m = \pm 1 \quad \text{takes sign of } \lambda \tag{8.36}$$

$$S_n = \pm 1 \quad \text{takes sign of } \phi \tag{8.37}$$

For the **Adams Hemisphere in a Square** projection where $|\lambda| \le \pi/2$:

$$\cos a = \cos\phi \sin\lambda \tag{8.38}$$
$$b = \frac{\pi}{2} - \phi \tag{8.39}$$
$$S_m = \pm 1 \quad \text{takes sign of } \sin\phi + a \tag{8.40}$$
$$S_n = \pm 1 \quad \text{takes sign of } \sin\phi - a \tag{8.41}$$

For the **Adams World in a Square I** poles at centers of sides projection:

$$\sin\phi' = \tan\frac{\phi}{2} \tag{8.42}$$
$$\cos a = \left(\cos\phi' \sin\frac{\lambda}{2} - \sin\phi'\right)/\sqrt{2} \tag{8.43}$$
$$\cos b = \left(\cos\phi' \sin\frac{\lambda}{2} + \sin\phi'\right)/\sqrt{2} \tag{8.44}$$
$$S_m = \pm 1 \quad \text{takes sign of } \lambda \tag{8.45}$$
$$S_n = \pm 1 \quad \text{takes sign of } \phi \tag{8.46}$$

For the **Adams World in a Square II** poles at opposite vertexes projection:

$$\sin\phi' = \tan\frac{\phi}{2} \tag{8.47}$$
$$\cos a = \cos\phi' \sin\frac{\lambda}{2} \tag{8.48}$$
$$\cos b = \sin\phi' \tag{8.49}$$
$$S_m = \pm 1 \quad \text{takes sign of } \sin\phi' + a \tag{8.50}$$
$$S_n = \pm 1 \quad \text{takes sign of } \sin\phi' - a \tag{8.51}$$

Finally compute:

$$\sin m = \pm(1 + \cos a \cos b - \sin a \sin b)^{\frac{1}{2}} \quad \text{where m takes the sign of } S_m \tag{8.52}$$
$$\sin n = \pm(1 - \cos a \cos b - \sin a \sin b)^{\frac{1}{2}} \quad \text{where n takes the sign of } S_n \tag{8.53}$$
$$x = F(m, \sqrt{0.5}) \tag{8.54}$$
$$y = F(n, \sqrt{0.5}) \tag{8.55}$$

where $F(\phi, k)$ is the elliptic integral of the first kind. Because the factor k is moderately large and because it is constant and the function itself is well behaved, the use of a Chebyshev approximation series is warranted.

$$F(\phi, \sqrt{0.5}) \approx \left[\sum_{i=0}^{N-1} c_i T_i(\phi)\right] - \frac{1}{2}c_0 \tag{8.56}$$

where

$$T_0(\phi) = 1 \tag{8.57}$$
$$T_1(\phi) = \phi \tag{8.58}$$
$$T_2(\phi) = 2\phi^2 - 1 \tag{8.59}$$
$$\cdots$$
$$T_{n+1}(\phi) = 2\phi T_n(\phi) - T_{n-1} \quad n \ge 1 \tag{8.60}$$

Normalizing the elliptic integral, $F(\phi, k)/\phi$ allows an even Chebyshev series to be determined with significantly fewer terms for a given precision. The follow list of even coefficients (stored in order) provide for an approximating function with a precision better than 1×10^{-7} which should be sufficient for spherical earth applications.

$$\begin{aligned} c_0 &= 2.19174570831038 & c_4 &= 5.30394739921063e-05 \\ c_1 &= 0.0914203033408211 & c_5 &= 3.12960480765314e-05 \\ c_2 &= -0.00575574836830288 & c_6 &= 2.02692115653689e-07 \\ c_3 &= -0.0012804644680613 & c_7 &= -8.58691003636495e-07 \end{aligned}$$

These are evaluated using Clenshaw's recursion in the following manner:

$$\begin{aligned} x &= \phi\frac{2}{\pi} \quad \text{scale argument range to } \pm 1 \\ x &= 2x^2 - 1 \quad \text{compensate argument for even series} \\ t_1 &= t_2 = 0 \end{aligned}$$

For $i = M - 1$ while $i > 0$ do

$$\begin{aligned} t &= t_1 \\ t_1 &= 2xt_1 - t_2 + c_i \\ t_2 &= t \\ i &= i - 1; \end{aligned}$$

where M is the order of the coefficient array. Finally compute

$$F(\phi, \sqrt{0..5}) = \phi\left(xt_1 - t_2 + \frac{1}{2}c_0\right)$$

8.1.8 Lagrange.

`+proj=lagrng +W= +lat_1=` Fig. 8.5 Ref. [17, p.]

The factor M is the ratio of the difference in longitude from the central meridian to the a circular meridian to 90°. Thus for $M = 1$ the hemisphere is in a circle and for $M = 2$ the world is in a circle. Factor ϕ_1 is the central latitude of the projection and forms a straight line parallel. If $|\phi| = \pi/2$ then

$$x = 0 \tag{8.61}$$

$$y = \pm 2 \quad \text{where } y \text{ takes the sign of } \phi. \tag{8.62}$$

otherwise

$$A_1 = \left(\frac{1 + \sin\phi_1}{1 - \sin\phi_1}\right)^{\frac{1}{2W}} \tag{8.63}$$

$$A = \left(\frac{1 + \sin\phi}{1 - \sin\phi}\right)^{\frac{1}{2W}} \tag{8.64}$$

$$V = A_1 A \tag{8.65}$$

$$C = (V + 1/V)/2 + \cos\frac{\lambda}{W} \tag{8.66}$$

$$x = \frac{2}{C}\sin\frac{\lambda}{W} \tag{8.67}$$

$$y = (V - 1/V)/V \tag{8.68}$$

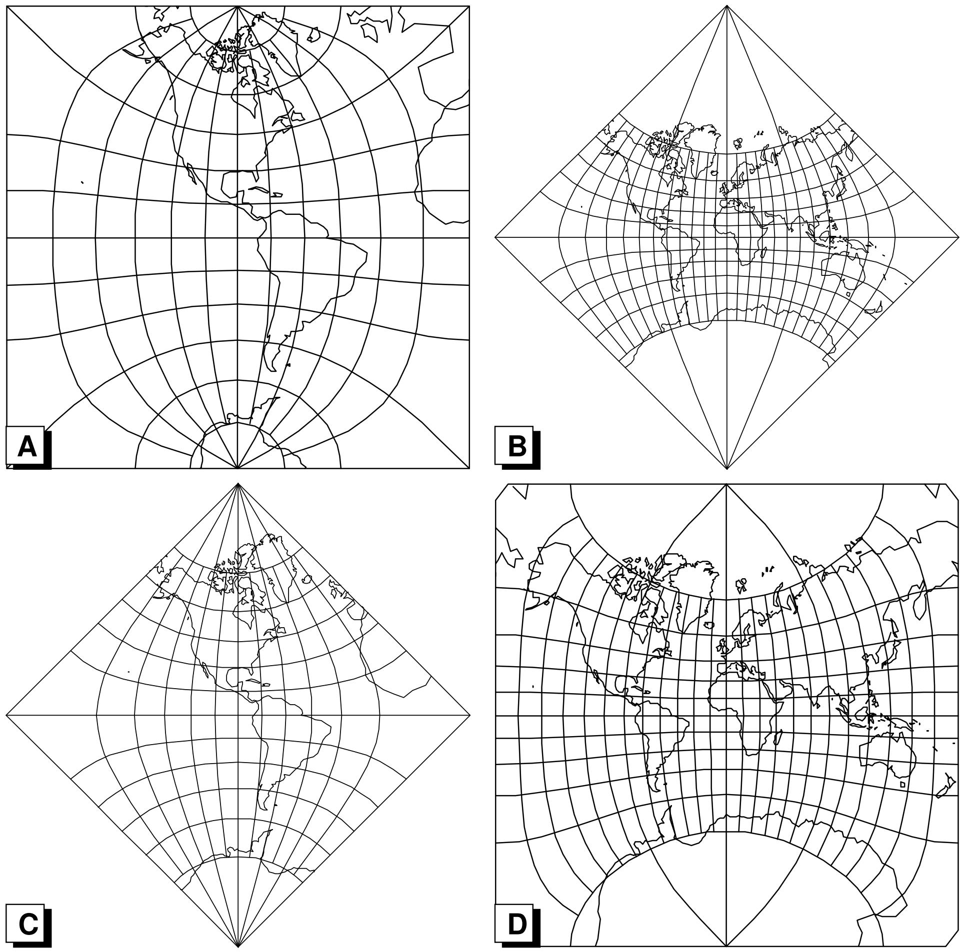

Figure 8.4: Miscellaneous Square Series
A–Guyou, **B**–Adams World in a Square I, **C**–Adams Hemisphere in a Square and **D**–Adams World in a Square II.

For normal Lagrange, $W = 2$ and $\phi_{=}0$ which are default values when omitted. If $W = 1$ and $\phi_1 = 0$ then equitorial Stereographic results.

8.1.9 Nicolosi Globular.

`+proj=nicol` Fig. 8.2 Ref. [17, p. 234]
If $\lambda = 0$ or $|\phi| = \pi/2$ then

$$x = 0 \qquad y = \phi \tag{8.69}$$

else if $\phi = 0$ then

$$x = \lambda \qquad y = 0 \tag{8.70}$$

else if $|\lambda| = \pi/2$ then

$$x = \lambda \cos\phi \qquad y = \frac{\pi}{2} \sin\phi \tag{8.71}$$

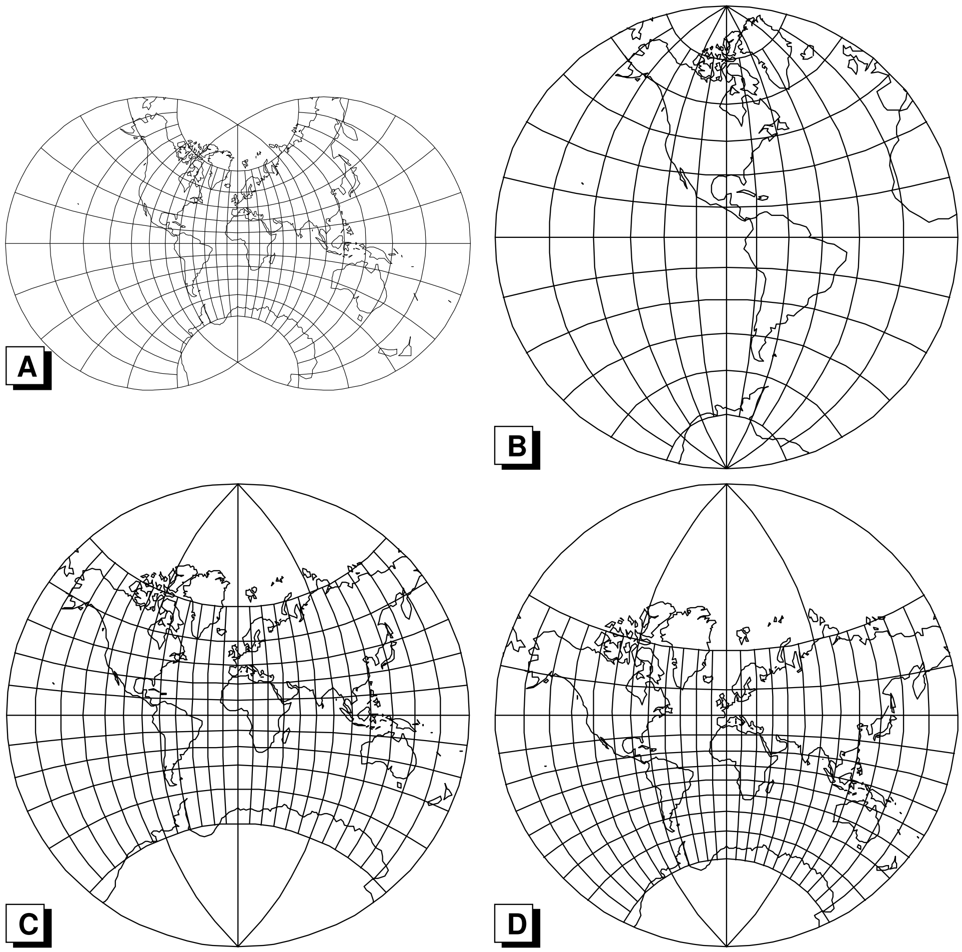

Figure 8.5: Lagrange Series
A–Lagrange `+W=1.43`, **B**–`+W=1 +lon_0=90W`, **C**–default options, **D**–`+lat_1=45N`.

else

$$b = \frac{\pi}{2\lambda} = \frac{2\lambda}{\pi} \tag{8.72}$$

$$c = \frac{2\phi}{\pi} \tag{8.73}$$

$$d = \frac{1 - c^2}{\sin\phi - c} \tag{8.74}$$

$$k = \frac{b}{d} \tag{8.75}$$

$$k_r = 1/k \tag{8.76}$$

$$M = \frac{k^2 \sin\phi + b/2}{1 + kr^2} \tag{8.77}$$

$$N = \frac{k_r^2 \sin\phi + d/2}{1 + k_r^2} \tag{8.78}$$

$$x = \frac{\pi}{2}\left(M \pm \left[M^2 + \frac{\cos^2\phi}{1 + k^2}\right]^{\frac{1}{2}}\right) \quad \text{where } \pm \text{ takes the sign of } \lambda \tag{8.79}$$

$$y = \frac{\pi}{2}\left(N \pm \left[N^2 - \frac{k_r^2 \sin^2\phi + d\sin\phi - 1}{1 + k_r^2}\right]^{\frac{1}{2}}\right) \tag{8.80}$$

where $\pm$ takes the opposite sign of ϕ

8.1.10 Van der Grinten (I).

`+proj=vandg` Fig. 8.6 Ref. [17, p. 237]

$$B = \frac{2}{\pi}|\phi| \qquad C = (1 - B^2)^{\frac{1}{2}} \tag{8.81}$$

If $\phi = 0$ then

$$x = \lambda \qquad y = 0 \tag{8.82}$$

else if $\lambda = 0$ then

$$x = 0 \qquad y = \pm\frac{\pi B}{1 + C} \tag{8.83}$$

else if $|\phi| = \pi/2$ then

$$x = 0 \qquad y = \pm\pi \tag{8.84}$$

where y takes the sign of ϕ in last two cases else

$$A = \left|\frac{\pi}{\lambda} - \frac{\lambda}{\pi}\right| \tag{8.85}$$

$$G = \frac{C}{B + C - 1} \tag{8.86}$$

$$P = G\left(\frac{2}{B} - 1\right) \tag{8.87}$$

$$Q = A^2 + B \tag{8.88}$$

$$S = P^2 + A^2 \tag{8.89}$$

$$T = G - P^2 \tag{8.90}$$

$$x = \pm\frac{\pi}{S}\left(AT + [A^2T^2 - S(G^2 - P^2)]^{\frac{1}{2}}\right) \tag{8.91}$$

$$y = \pm\frac{\pi}{S}\left(PQ - A[(A^2 + 1)S - Q^2]^{\frac{1}{2}}\right) \tag{8.92}$$

where x and y take the respective signs of λ and *phi*.

8.1.11 Van der Grinten II.

`+proj=vandg2` Fig. 8.6 Ref. [17, p. 237–238]

$$B = \frac{2}{\pi}|\phi| \qquad C = (1 - B^2)^{\frac{1}{2}} \tag{8.93}$$

If $\phi = 0$ then

$$x = \lambda \qquad y = 0 \tag{8.94}$$

else if $\lambda = 0$ then

$$x = 0 \qquad y = \pm\frac{\pi B}{1 + C} \tag{8.95}$$

else if $|\phi| = \pi/2$ then

$$x = 0 \qquad y = \pm\pi \tag{8.96}$$

where y takes the sign of ϕ in last two cases else

$$A = \left|\frac{\pi}{\lambda} - \frac{\lambda}{\pi}\right| \tag{8.97}$$

$$x_1 = \frac{C(1+A^2)^{\frac{1}{2}} - AC^2}{1+A^2B^2} \tag{8.98}$$

$$x = \pm\pi x_1 \tag{8.99}$$

$$y = \pm\pi(1 - x_1(2A + x_2))^{\frac{1}{2}} \tag{8.100}$$

where x and y take the respective signs of λ and phi.

8.1.12 Van der Grinten III.

`+proj=vandg3` Fig. 8.6 Ref. [17][p. 238], [13][p. 78]

$$B = \frac{2}{\pi}|\phi| \qquad C = (1 - B^2)^{\frac{1}{2}} \tag{8.101}$$

If $\phi = 0$ then

$$x = \lambda \qquad y = 0 \tag{8.102}$$

else if $\lambda = 0$ then

$$x = 0 \qquad y = \pm\frac{\pi B}{1+C} \tag{8.103}$$

else if $|\phi| = \pi/2$ then

$$x = 0 \qquad y = \pm\pi \tag{8.104}$$

where y takes the sign of ϕ in last two cases else

$$A = \left|\frac{\pi}{\lambda} - \frac{\lambda}{\pi}\right| \qquad y_1 = \frac{B}{1+C} \tag{8.105}$$

$$x = \pm((A^2 + 1 - y_1^2)^{\frac{1}{2}} - A \qquad y = \pm\pi y_1 \tag{8.106}$$

where x and y take the respective signs of λ and phi.

8.1.13 Van der Grinten IV.

`+proj=vandg4` Fig. 8.6 Ref. [17, p. 236]
If $\phi = 0$ then

$$x = \lambda \qquad y = 0 \tag{8.107}$$

else if $\lambda = 0$ or $|\phi| = \pi/2$ then

$$x = 0 \qquad y = \phi \tag{8.108}$$

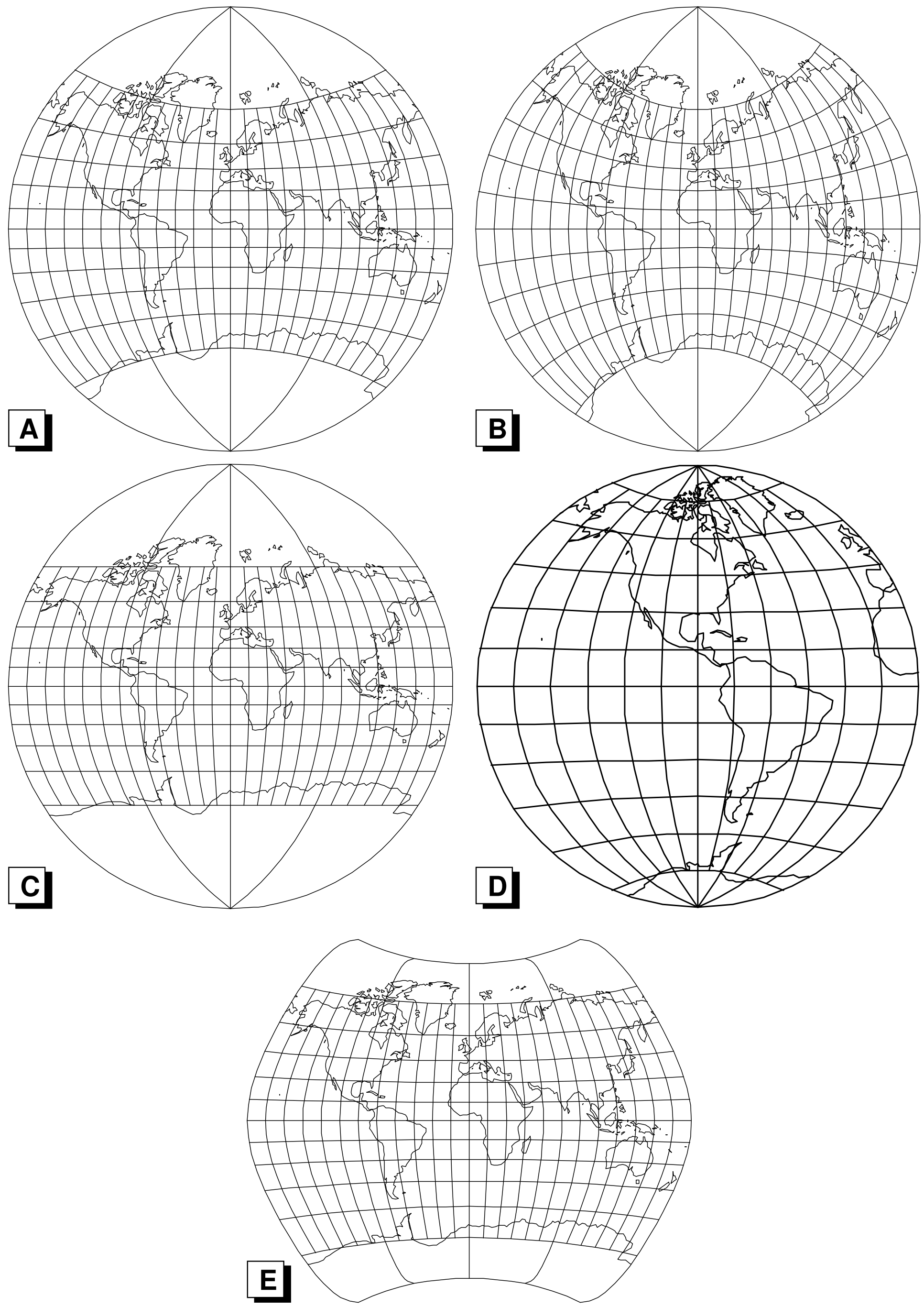

Figure 8.6: Van der Grinten Series
A–Van der Grinten (I), **B**–Van der Grinten II, **C**–Van der Grinten III, **D**–Van der Grinten IV and **E**–Larrivée.

else

$$B = \frac{2}{\pi}|\phi| \tag{8.109}$$

$$C = \frac{-5 + B(8 - B(2 + B^2))}{2B^2(B-1)} \tag{8.110}$$

$$R = \frac{2\lambda}{\pi} \tag{8.111}$$

$$D = \pm\left\{\left(R + \frac{1}{2}\right)^2 - 4\right\}^{\frac{1}{2}} \quad \text{taking the sign of } (\lambda - \pi/2) \tag{8.112}$$

$$F = (B + C)^2(B^2 + C^2D^2 - 1) + (1 - B^2) \tag{8.113}$$

8.1.14 Larrivée.

`+proj=larr` Fig. 8.6 Ref. [15][p. 262]
Similar to Van der Grinten I but without circular arcs.

$$x = \lambda\left(1 + \sqrt{\cos\phi}\right)/2 \qquad y = \phi/\left(\cos\frac{\phi}{2}\cos\frac{\lambda}{6}\right) \tag{8.117}$$

Chapter 9

Oblique Projections

All of the spherical forms of the **libproj4** projection library can be used as an oblique projection by means of the `pj_translate` function described on . The user performs the oblique transformation by selecting the oblique projection `+proj=ob_tran`, specifying the translation factors, `o_lat_p=`α, and `o_lon_p=`β, and the projection to be used, `+o_proj=`*proj*. In the example of the Fairgrieve projection the latitude and longitude of the pole of the new coordinates, α and β respectively, are to be placed at 45°N and 90°W and use the Mollweide projection. Beccause the central meridian of the translated coordinates will follow the β meridian it is necessary to translate the the translated system so that the Greenwich meridian will pass through the center of the projection by offsetting the central meridian. The final control for this projection is:

```
+proj=ob_tran +o_proj=moll +o_lat_p=45 +o_lon_p=-90 +lon_0=-90
```

Figure **??** shows a plot the resultant projection. Two more examples of oblique Mollweide projections are shown in figure **??**.

9.0.15 Oblique Projection Parameters From Two Control Points

A convenient method of determining the position of the translated pole is by specifying two points along the central meridian with the equations:

$$\beta = \arctan\left(\frac{\cos\phi_1 \sin\phi_2 \cos\lambda_1 - \sin\phi_1 \cos\phi_1 \cos\lambda_1}{\sin\phi_1 \cos\phi_2 \sin\lambda_2 - \cos\phi_1 \sin\phi_2 \sin\lambda_1}\right) \tag{9.1}$$

$$\alpha = \arctan\left(\frac{-\cos(\beta - \lambda_1)}{\tan\phi_1}\right) \tag{9.2}$$

References

[1] *Formulas and constants for the calculation of the Swiss conformal cylindrical projection and for the transformation between coordinate systems*. Switzerland, 2001.

[2] Guidance note number 7: Coordinate conversions and transformations including formulas. Technical report, European Petrolium Survey Group, 2004.

[3] John P. Snyder Alan A. DeLucia. An innovative world map projection, 1986.

[4] Paul B. Anderson. Personal communications, 2004.

[5] János Baranyi. The problems of the representation of the globe on a plane with special reference to the preservation of the forms of the continents. In *Hungarian Cartographical Studies*, pages 19–43. Földmérési Intézet, Budapest, 1968.

[6] Frank Canters. *Small-scale Map Projection Design*. Taylor & Francis, London, 2002.

[7] Martin Hotine. The orthomorphic projection of the spheroid. *Empire Survey Review*, 8(62):300–311, 1946.

[8] D. H. Maling. *Coordinate Systems and Map Projections*. Pergamon Press, New York, second edition, 1992.

[9] Philip M.Voxland. Personal communications, 2004.

[10] Frederick Pearson II. *Map Projections: Theory and Applications*. CRC Press, Boca Raton, Florida, 1990.

[11] Author H. Robinson. A new map projection: Its development and characteristics. *International Yearbook of Cartography*, 14:145–155, 1974.

[12] Henri Roussihle. Emploi des coordonnées rectangulaires stéréographiques pour le calcul de la triangulation dans un rayon de 560 kilomètres autour de l'origine. Travaux, International Union of Geodesy and Geophysics, May 1922.

[13] John P. Snyder. A comparison of pseudocylindrical map projections. *The American Cartographer*, 4(1):60–81, April 1977.

[14] John P. Snyder. Map projections—a working manual. Prof. Paper 1395, U.S. Geol. Survey, 1987.

[15] John P. Snyder. *Flattening of the Earth—Two Thousand Years of Map Projections*. Univ. of Chicago Press, Chicago and London, 1993.

[16] John P. Snyder. The hall eucyclic projection. *Cartography and Geographic Information Systems*, 21(4):213–218, 1994.

[17] John P. Snyder and Philip M. Voxland. An album of map projections. Prof. Paper 1453, U.S. Geol. Survey, 1989.

[18] Paul D. Thomas. Conformal projections in geodesy and cartography. Spec. Pub. 251, U.S. Coast and Geodetic Survey, 1952.

[19] W. R. Tobler. The hyperelliptical and other new pseudo cylindrical equal area map projections. *Journal of Geophysical Research*, 78(11):1753–1759, April 1973.

[20] U.S. Geological Survey. L176—batch general map projection transform. NMD User's Manual, U.S. Geol. Survey, 1989.

[21] U.S. Geological Survey. GCTP—general cartographic transformation package. NMD Software Documentation, U.S. Geol. Survey, 1990.

[22] Karlheinz Wagner. Kartographische netzentwürfe. Technical report, Bibliographisches Institut Leipzig, 1949.

Index