forSISL

Modern Fortran Interfaces to the SINTEF Spline Library (SISL)

Users Manual (Draft)

Version 1.0 (Initial Github Release)

March 7, 2021

Prepared by

Richard Weed, Ph.D.

rweedmsu@gmail.com

1. Introduction

The forSISL library provides Modern Fortran C-Interoperability interfaces to the SINTEF Spline Library (SISL) C library for NURBS curve and surface modeling and interrogation. These interfaces allow Fortran programmers to call the underlying SISL Library C routines in an almost seamless manner without the traditional C based wrappers previously used to interface C and Fortran. Users who are unfamiliar with the Fortran C-interoperability introduced in the Fortran 2013 standard and modified in subsequent standards (2008 and 2018) should obtain a copy of a recent Fortran programming reference such as *Modern Fortran Explained* by Metcalf, Reid, and Cohen, Oxford University Press, 2018 before attempting to use this software.

With the exception of a handful of graphics driver routines designed to be replaced by user supplied versions of the functions, all of the C functions described version 4.4 of the SISL Reference Manual that is supplied with the SINTEF SISL library software are supported (a total of 173 routines). In addition, a set of routines to write and read SISL surface and curve objects to/from Fortran sequential formatted files are provided. These mimic the SINTEF Go software format C++ routines found in the SISL C library software. The user is warned however that only a small number of the 173 routines have been tested at this time. The routines that have been tested are used in the fifteen example programs that are supplied with the forSISL source. Each of the example programs are Fortran implementations of the examples supplied with the SINTEF C library software. These examples have all been tested and the results verified to match the output generated by their C++ equivalents.

A complete verification of all of the 173 routines in the forSISL library will require the development of a substantial number of unit tests that exercise each function. A complete and thorough V&V of the entire code base is planned but will require a substantial expenditure in time and effort so it is delayed for future releases. However, since the majority of the SISL C functions share a lot of their signature (interface) with other functions, it is felt that the probability of the functions that have not been tested will work properly is high. Users are advised and encouraged however to treat the initial release of forSISL as early beta code and report any problems to the developer.

The SISL C library software is available from https://github/SINTEF-Geometry/SISL. Users must build the SISL C library and link it into their projects prior to attempting to use the forSISL interfaces.

2.0 Overview of the forSISL software

2.1 Issues related to interoperability of data types

The forSISL software attempts to match the functionality of the SISL C routines as closely as possible. Each subroutine interface was written to keep the same variable names and function signature as their C equivalents with a few exceptions. The SISL C routines take in integer and double precision data along with previously defined SISL curve, surface, and interface curve objects (C structures) and return a curve, surface, or interface object along with double precision (C double) arrays and other scalar data. In some instances, arrays of the structures are returned.

For arrays of data, SISL assumes that memory for the arrays is provided in in two ways: 1) the user allocates memory for them prior to calling the SISL function and 2) the SISL function allocates memory inside the function and returns a pointer to the array. For the first case, arrays are passed into and out of the Fortran routines using Fortran assumed size arrays. These arrays are directly inter-operable with C unknown size arrays defined by array names with empty brackets (ie A[]). For memory allocated inside the SISL C functions, SISL returns pointers and sometimes multiple levels of indirection ie **A etc). In the Fortran interfaces, these arrays are are copied into Fortran ALLOCATABLE arrays and the original C arrays are freed.

The SISL C curve, surface and interface curve objects contain a variety of components including pointers to other objects. The following approach to handling the C structure objects in as seamless manor as possible in Fortran was used:

1. Fortran C binding versions of the equivalent C structures were created Where the C structures contained components with pointers with multiple indirections, a single variable of type C_PTR was used to represent the equivalent C variable. A single C_PTR data type was used to represent Real arrays etc. Other data was defined as C_INT for scalar integers and C_DOUBLE for real values. This approach allows the Fortran C bound derived type to interact directly with its C equivalent

2. Although the Fortran C bound type can interact directly with its C equivalent, it was decided to provide additional Fortran derived types that mirror the C structures but provide a mechanism for accessing any arrays and other objects directly without having to constantly unroll the C_PTR variables prior to use. A mapping routine was written for each object that takes the object pointer returned from a call to a SISL C function and performs the following operations.

a) All scalar data (integers etc) are copied to equivalent Fortran variables

b) All arrays of real data are mapped via the C_F_POINTER function to Fortran array pointers

c) All object components in the C structures are replaced by Fortran equivalents and their scalar and array contents are also either copied or mapped to Fortran array pointers.

d) A single C_PTR variable component was added to the Fortran structure to hold the instance of the original object pointer returned by a SISL C function. This provides a seamless way of returning the pointer to subsequent calls of the SISL functions and most importantly provides a way of returning the C object pointer to the SISL routines designed to free any memory allocated internally by the C functions. The one cardinal rule of Fortran-C interoperability is the processor (C or Fortran) that allocates the memory must be the one that frees or deallocates it.

e) The Fortran routines that wrap the C free functions will in addition NULLIFY all associated Fortran pointers etc prior to calling the C free function.

For a more detailed understanding of how the interoperability interfaces work users are advised to study the definition of the derived types and the mapping routines given in module

forSISLdata.F90 and the other routines. Users will also have to consult the SISL reference manual for detailed definitions of the argument list variables, how they are passed, and what their memory allocation requirements are.

Descriptions of the Fortran and C versions of the SISL curve, surface, and interface curve objects are given in Appendix A. A complete list of the current Fortran routine interfaces is given in Appendix B. For definitions of the variables in the interfaces, see the SISL library reference manual.

2.2 Overview of the code base

The forSISL code base consists of twelve files that each contain a single Fortran module. The files with names beginning with Curve or Surface contain the interfaces defined in the SISL reference manual chapters with titles equivalent to the file name. Example: Curve_Definition.F90 contains the interfaces defined in Chapter 2 of the SISL reference manual. forSISLdata.F90 contains the definitions of the SISL objects (SISLcurve, SISLsurf, SISLIntcurve etc) and the mapping routines. forSISLio.F90 contains a small set of read and write routines for inputting and outputting the curve and surface objects. The remaining forSISL files are convenience modules group the other modules together to provide access to either the curve or surface modules as group or all of the modules except forSISLdata.F90 with one USE statement. A small C routine is also provided that contains a wrapper around the C free function.

3.0 Building forSISL

forSISL is currently only supported on Linux systems but building on an Apple Mac should be straight forward. There are no current plans to support Windows but; due to the relatively small number of files that make up forSISL, creating a Visual Studio project should also be straight forward. The forSISL software had been compiled and tested with GNU GCC gfortran versions 9 and 10 and the Intel oneAPI compilers (ifort). The forSISL library does not depend on the SISL C library to build correctly but requires it to operate so users should download and build the SISL C library first and study the reference manual prior to building and using forSISL. A simple build facility based on standard Unix/Linux make files is provided but users could just as easily compile the code with a simple script (making sure the modules are compiled in the proper order). A configure bash script is provided that will set environment variables and Makefile macros for specific compilers and various commonly used compiler options. By default O2 optimization is assumed along with modern Fortran allocation on assignment. A successful build will copy all module .mod files to the forSISL module directory and build archive and shared object libraries named forSISL.a and forSISL.so and place them in the forSISL lib subdirectory. Future versions will include a make install feature to allow users to place the library and the module files in alternate locations. The steps to build forSISL and the fifteen example codes are as follows.

1. cd to the forSISL directory and run the configure script (note you must preface configure with either source or a . under the bash shell. I use source in these examples). Also you compiler of choice must be visible in your default path. Also by default, the configure script will use whatever version of gfortran that is visible as /usr/lib/gfortran etc. as the default if no compiler is specified

Example 1. Build forSISL with the Intel compiler and the default O2 options. Other compiler options are defined in the /config subdirectory files

source ./configure –compiler=intel
make cleanall
make

Example 2. Build forSISL with Intel all warnings turned on and zero optimization

source ./configure –compiler=intel –warn=all –opt-level=0
make cleanall
make

Example 3. Some Linux systems allow different versions of gfortran to be installed with the versions differentiated by the addition of the version number on the file name. Version 10 of GCC compilers will be installed as /usr/bin/gcc-10 and /usr/bin/gfortran-10. The configure script is set up to allow users with multiple versions of gcc compilers with the above naming convention to be used by specifying one of gcc8, gcc9, or gcc 10 as the compiler option

source ./configure –compiler=gcc9 –opt-level=0
make cleanall
make

Other configure options may be added in the future.

2 Once the forSISL libraries are built you can then build all of the example programs by:

make SISL_LIBPATH=/your/pathto/SISL/libs examples

Alternatively, there are simple scripts in the example directory for building the examples one at a time for the intel, gcc 9, and gcc 10 compilers. To use them you need to first export the path to where the SISL C libraries live. Be aware though that all the example programs require input files generated by the previous examples so you should run the examples in sequence (1-15).

export SISL_LIBPATH=/your/pathto/SISL/libs
makex_linux example04

Note these scripts expect the SISL C library files to be named sisl.a and sisl.so

There are several clean options that will remove .o, .mod files etc.

make clean will just remove .o and .mod files from the src directory

make cleanall removes all .o and .mod files from the src directory, all .mod files from the module directory, .a and .so files from the lib directory, and all example.x files and the .g2 files generated by the example programs.

4.0 Using the forSISL library

To use the forSISL library you must link your program with either the forSISL.a or forSISL.so files and the equivalent SISL C libraries. The example programs should be studied for a more detailed description of how the Fortran interfaces are used but a simple example would be a call to SISL routine s1356 that generates a SISL spline curve that interpolates a set of given points. The following is taken from example02.F90 and illustrates the required USE associations, data types, declaring an ALLOCATABLE array to hold an array returned by SISL C version of s1356 as a pointer, using the output routines to write out the points and curve data, and freeing and cleaning up memory before exiting the program.

```
Program example02

!  This program will generate a SISL spline curve object by
!  interpolating a set of given points.

  USE forSISLdata ! Brings in REAL64 and SISLcurve
  USE forSISL,    ONLY: s1356, freeCurve, writeSISLPoints, writeSISLcurve

  Implicit NONE

  Integer                 :: num_points, jnbpar, jstat, os_points,   os_curve
  Real(REAL64)            :: cstartpar, cendpar
  Integer                 :: itype(6)
  Real(REAL64)            :: points(18)
  Real(REAL64),     ALLOCATABLE :: gpar(:)
  Type(SISLcurve)         :: result_curve
  Character(LEN=:), ALLOCATABLE :: OUT_FILE_POINTS
  Character(LEN=:), ALLOCATABLE :: OUT_FILE_CURVE

  OUT_FILE_POINTS = "example02_points.g2"
  OUT_FILE_CURVE  = "example02_curve.g2"
  num_points = 6
  cstartpar  = 0.0_REAL64
  itype      = [1, 1, 1, 1, 1, 1 ]
  points     = REAL( [0, 0, 0,                              &
                      1, 1, 0,                              &
                      2, -1, 0,                             &
                      3, 0, 0,                              &
                      4, 1, 1,                              &
                      3, 0, 4], REAL64)

! Generate curve that interpolates points

  Call s1356(points,        &! pointer to where the point coordinates are stored
             num_points,    &! number of points to be interpolated
             3,             &! the dimension
             itype,         &! what type of information is stored at a particular point
             0,             &! no additional condition at start point
```

```
            0,            &! no additional condition at end point
            1,            &! open curve
            4,            &! order of the spline curve to be produced
            cstartpar,    &! parameter value to be used at start of curve
            cendpar,      &! parameter value at the end of the curve (to be determined)
            result_curve, &! the resulting spline curve (to be determined)
            gpar,         &! array of parameter values of the points in the curve
                          ! (to be determined)
            jnbpar,       &! number of unique parameter values (to be determined)
            jstat)        ! status message


  Print *,''
  Write (OUTPUT_UNIT, '(" Total parameter interval of curve : [ ",g0.7," ", g0.7," ]")') cstartpar,
cendpar
  Print *,''
  Write(OUTPUT_UNIT, '(" Point parameter values was decided to be: ")')
  Write(OUTPUT_UNIT, '(80(g0.7," "))') gpar(:)
  Open (newunit=os_points, FILE=OUT_FILE_POINTS, FORM="FORMATTED",          &
       STATUS="UNKNOWN")
  Open (newunit=os_curve,  FILE=OUT_FILE_CURVE,  FORM="FORMATTED",          &
       STATUS="UNKNOWN")

! output curve and points

  Call writeSISLPoints(num_points, points, os_points)
  Call writeSISLCurve( result_curve, os_curve)

! free curve and gpar

  Call freeCurve(result_curve)
  If (ALLOCATED(gpar)) DEALLOCATE(gpar)

! Make valgrind happy with gfortran 9 which doesn't allocate these on exit as it should
  If (ALLOCATED(OUT_FILE_POINTS)) DEALLOCATE(OUT_FILE_POINTS)
  If (ALLOCATED(OUT_FILE_CURVE))  DEALLOCATE(OUT_FILE_CURVE)

  Close(os_points)
  Close(os_curve)

  STOP

End Program example02
```

5.0 Testing and Known issues

As stated previously, only a small fraction of the 170 plus C interfaces have been tested so until a complete set of unit tests can be written this software should be considered to be an

early beta release. However, I have a fairly high level of confidence that the other interfaces that I haven't tested (as long as there aren't hidden typos that my several reviews of the source haven't caught) will work because I was basically duplicating code from function to function. One issue that came up in initial testing is a memory leak that valgrind caught in example13 that I have yet to track down. It doesn't appear when you run the C++ version in the SISL C libraries so it's related to Fortran somehow. All the other examples pass valgrind tests with no leaks. With the exception of example 15, all of the tests generated results that are equivalent (to the precision that the data is written to the output files) to the SISL C library results. Example 15 implements two simple ray tracing routines to compute intersections of a straight line with a B-spline surface. The "robust" ray tracer output generated by the Fortran interfaces is the same as the C version. The Fortran version of the alternate "quick" ray tracer produces five fewer points than the C version. My current working theory is that the Fortran files generated by the previous examples (10 and 11) that are read by example 15 have floating data that is written with more significant digits than the C++ example 15. I think this affects the accuracy of some of the intersection tests.

Another potential issue is with how intersection curves are generated and freed. When the newIntCurve function is used to create an instance of an SISLintCurve object pointers to arrays passed into the C function are set and stored in the object instead of making a copy of the arrays and setting pointers to the copies. The problem is that a subsequent call to freeIntCurve will attempt to free the original arrays via the pointers. This raises the possibility with the Fortran interfaces that the underlying C freeIntCurve function will try to free arrays defined in Fortran since the arrays are passed directly from Fortran to C. Note this problem does not occur with instance of SISLIntCurve generated by the other C functions, only with instances created with newIntCurve. I think I know a way around this but have not had time to implement it and test it out.

6.0 Support

Questions and bug reports can be posted as issues on the github site. They can also be addressed to me via email at rweedmsu@gmail.com.

7.0 Summary and Future Plans

A library of Modern Fortran C-interoperability interfaces to the SINTEF Spline Library (SISL) has been developed. However, the initial release has not undergone the kind of extensive testing required to verify that all 170 plus interfaces work correctly. An extensive suite of unit tests that will test all of the interfaces is planned for future releases as well as improved documentation. The current release has undergone initial testing using Fortran implementations of the fifteen example programs that are part of the SISL C library. Users are again advised to treat this code as a very early beta release and not something you want to include in production code at this point. It is being released in this state to encourage other pairs of eyes to review the code.

Appendix A. SISL Structure Object Overview

A.1 Description of forSISL and SISL structure objects

The following describes the SISL curve, surface, interface curve, bounding box, direction cone and segmentation objects for both two Fortran versions of the objects used by the forSISL interfaces and the equivalent C object.

A.2 The SISLCurve objects

The two SISLCurve objects defined in Fortran and the original C structure have the following formats

A.2.1 Fortran C binding object

```
Type, BIND(C) :: c_SISLCurve  ! C curve object
   Integer(C_INT) :: ik     = 0_C_INT
   Integer(C_INT) :: in     = 0_C_INT
   Type(C_PTR)    :: et     = C_NULL_PTR
   Type(C_PTR)    :: ecoef  = C_NULL_PTR
   Type(C_PTR)    :: rcoef  = C_NULL_PTR
   Integer(C_INT) :: ikind  = 0_C_INT
   Integer(C_INT) :: idim   = 0_C_INT
   Integer(C_INT) :: icopy  = 0_C_INT
   Type(C_PTR)    :: pdir   = C_NULL_PTR
   Type(C_PTR)    :: pbox   = C_NULL_PTR
   Integer(C_INT) :: cuopen = 0_C_INT
 End Type
```

A.2.2 Fortran Interface object

```
Type :: SISLCurve   ! Fortran curve object
   Integer                :: ik        = 0
   Integer                :: in        = 0
   Real(REAL64), Pointer  :: et(:)     => NULL()
   Real(REAL64), Pointer  :: ecoef(:)  => NULL()
   Real(REAL64), Pointer  :: rcoef(:)  => NULL()
   Integer                :: ikind     = 0
   Integer                :: idim      = 0
   Integer                :: icopy     = 0
   Type(SISLdir)          :: pdir
   Type(SISLbox)          :: pbox
```

```
    Integer                 :: cuopen    = 0
    Type(C_PTR)             :: cptr      = C_NULL_PTR
  End Type
```

A.2.3 SISL C structure (from SISL library) with variable definitions

```
typedef struct SISLCurve
{
  int ik;           /* Order of curve.                          */
  int in;           /* Number of vertices.                      */
  double *et;       /* Pointer to the knotvector.               */
  double *ecoef;    /* Pointer to the array containing vertices. */
  double *rcoef;    /* Pointer to the array of scaled vertices if
                       rational.  */
  int ikind;        /* Kind of curve
                       = 1 : Polynomial B-spline curve.
                       = 2 : Rational B-spline curve.
                       = 3 : Polynomial Bezier curve.
                       = 4 : Rational Bezier curve.            */
  int idim;         /* Dimension of the space in which the curve
                       lies.      */
  int icopy;        /* Indicates whether the arrays of the curve
                       are copied or referenced by creation of the
                       curve.
                       = 0 : Pointer set to input arrays.
                       = 1 : Copied.
                       = 2 : Pointer set to input arrays,
                       but are to be treated as copied.   */
  SISLdir *pdir;    /* Pointer to a structur to store curve
                       direction.      */
  SISLbox *pbox;    /* Pointer to a structur to store the
                       surrounded boxes. */
  int cuopen;       /* Open/closed flag.                        */
} SISLCurve;
```

A.3 The SISLSurf Objects

The two SISLSurf objects defined in Fortran and the original C structure have the following formats

A.3.1 Fortran C binding object

```
Type, BIND(C) :: c_SISLsurf   ! C surf object
  Integer(C_INT) :: ik1       = 0_C_INT
  Integer(C_INT) :: ik2       = 0_C_INT
  Integer(C_INT) :: in1       = 0_C_INT
  Integer(C_INT) :: in2       = 0_C_INT
  Type(C_PTR)    :: et1       = C_NULL_PTR
  Type(C_PTR)    :: et2       = C_NULL_PTR
  Type(C_PTR)    :: ecoef     = C_NULL_PTR
  Type(C_PTR)    :: rcoef     = C_NULL_PTR
  Integer(C_INT) :: ikind     = 0_C_INT
  Integer(C_INT) :: idim      = 0_C_INT
  Integer(C_INT) :: icopy     = 0_C_INT
  Type(C_PTR)    :: pdir      = C_NULL_PTR
  Type(C_PTR)    :: pbox      = C_NULL_PTR
  Integer(C_INT) :: use_count = 0_C_INT
  Integer(C_INT) :: cuopen_1   = 0_C_INT
  Integer(C_INT) :: cuopen_2   = 0_C_INT
  Type(C_PTR)    :: seg1      = C_NULL_PTR
  Type(C_PTR)    :: seg2      = C_NULL_PTR
  Integer(C_INT) :: sf_type   = 0_C_INT
End Type
```

A.3.2 Fortran Interface Object

```
Type :: SISLsurf   ! Fortran surf object
   Integer                 :: ik1      = 0
   Integer                 :: ik2      = 0
   Integer                 :: in1      = 0
   Integer                 :: in2      = 0
   Real(REAL64), Pointer   :: et1(:)   => NULL()
   Real(REAL64), Pointer   :: et2(:)   => NULL()
   Real(REAL64), Pointer   :: ecoef(:) => NULL()
   Real(REAL64), Pointer   :: rcoef(:) => NULL()
   Integer                 :: ikind    = 0
   Integer                 :: idim     = 0
   Integer                 :: icopy    = 0
   Type(SISLdir)           :: pdir
   Type(SISLbox)           :: pbox
   Integer                 :: use_count = 0
   Integer                 :: cuopen_1  = 0
   Integer                 :: cuopen_2  = 0
   Type(SISLSegmentation)  :: seg1
   Type(SISLSegmentation)  :: seg2
   Integer                 :: sf_type  = 0
   Type(C_PTR)             :: cptr  = C_NULL_PTR
 End Type
```

A.3.3 SISL C structure (from SISL library) with variable definitions

```
typedef struct SISLSurf
{
  int ik1;                      /* Order of surface in first parameter
                                   direction.       */
  int ik2;                      /* Order of surface in second parameter
                                   direction.      */
  int in1;                      /* Number of vertices in first parameter
                                   direction.     */
  int in2;                      /* Number of vertices in second parameter
                                   direction.    */
  double *et1;                  /* Pointer to knotvector in first parameter
                                   direction.  */
  double *et2;                  /* Pointer to knotvector in second parameter
                                   direction. */
  double *ecoef;                /* Pointer to array of vertices of surface. */
  double *rcoef;                /* Pointer to the array of scaled vertices
                                   if surface is rational. */
  int ikind;                    /* Kind of surface
                                   = 1 : Polynomial B-spline tensor-product
                                         surface.
                                   = 2 : Rational B-spline tensor-product
                                         surface.
                                   = 3 : Polynomial Bezier tensor-product
                                         surface.
                                   = 4 : Rational Bezier tensor-product
                                         surface.                          */
  int idim;                     /* Dimension of the space in which the surface
                                   lies.    */
  int icopy;                    /* Indicates whether the arrays of the surface
                                   are copied or referenced by creation of
                                   the surface.
                                   = 0 : Pointer set to input arrays.
                                   = 1 : Copied.
                                   = 2 : Pointer set to input arrays,
```

```
                                                but are to be treated as copied.
*/
  SISLdir *pdir;                /* Pointer to a structur to store surface
                                   direction.    */
  SISLbox *pbox;                /* Pointer to a structur to store the
                                   surrounded boxes. */
  int use_count;                /* use count so that several tracks can share
                                   surfaces, no internal use */
 int cuopen_1;                  /* Open/closed flag, 1. par directiion */
 int cuopen_2;                  /* Open/closed flag. 2. par direction  */
  SISLSegmentation *seg1;       /* Segmentation information for use in
                                   intersection functionality, 1. par. dir. */
  SISLSegmentation *seg2;       /* Segmentation information for use in
                                   intersection functionality, 2. par. dir. */
  int sf_type;        /* SURFACE_TYPE, Flag for special surface information */
} SISLSurf;
```

A.4 The SISLIntcurve Objects

The two SISLIntCurve objects defined in Fortran and the original C structure have the following formats

A.4.1 Fortran C binding object

```
 Type, BIND(C) :: c_SISLIntCurve ! C interface curve object
    Integer(C_INT) :: ipoint    = 0_C_INT
    Integer(C_INT) :: ipar1     = 0_C_INT
    Integer(C_INT) :: ipar2     = 0_C_INT
    Type(C_PTR)    :: epar1     = C_NULL_PTR
    Type(C_PTR)    :: epar2     = C_NULL_PTR
    Type(C_PTR)    :: pgeom     = C_NULL_PTR
    Type(C_PTR)    :: ppar1     = C_NULL_PTR
    Type(C_PTR)    :: ppar2     = C_NULL_PTR
    Integer(C_INT) :: itype     = 0_C_INT
    Integer(C_INT) :: pretop(4) = [0_C_INT, 0_C_INT, 0_C_INT, 0_C_INT]
  End Type
```

A.4.2 Fortran Interface object

```
 Type :: SISLIntCurve   ! Fortran interface curve object
    Integer                  :: ipoint    = 0
    Integer                  :: ipar1     = 0
    Integer                  :: ipar2     = 0
    Real(REAL64),    Pointer :: epar1(:)  => NULL()
    Real(REAL64),    Pointer :: epar2(:)  => NULL()
    Type(SISLCurve)          :: pgeom
    Type(SISLCurve)          :: ppar1
    Type(SISLCurve)          :: ppar2
    Integer                  :: itype     = 0
    Integer                  :: pretop(4) = [0, 0, 0, 0]
    Type(C_PTR)              :: cptr = C_NULL_PTR
  End Type
```

A.4.3 SISL C structure (from SISL library) with variable definitions

```
typedef struct SISLIntcurve
{
  int ipoint;                  /* Number of points defining the curve.      */
  int ipar1;                   /* Number of parameter directions of first
                                  object.                                   */
  int ipar2;                   /* Number of parameter directions of second
                                * object.                                   */
  double *epar1;               /* Pointer to the parameter-values of the
                                  points
                                  in the first object.                      */
  double *epar2;               /* Pointer to the parameter-values of the
                                  points
                                  in the second object. If one of the objects
                                  is an analytic curve or surface epar2 points
                                  to nothing.                               */
  SISLCurve *pgeom;            /* Pointer to the intersection curve in the
                                  geometry space. If the curve is not
                                  computed, pgeom points to nothing.       */
  SISLCurve *ppar1;            /* Pointer to the intersection curve in the
                                  parameter plane of the first object. If
```

```
                                   the curve is not computed, ppar1 points
                                   to nothing.                              */
  SISLCurve *ppar2;              /* Pointer to the intersection curve in the
                                   parameter plane of the second object. If
                                   the curve is not computed, ppar2 points
                                   to nothing.                              */
  int itype;                     /* Kind of curve.
                                   = 1 : Straight line.
                                   = 2 : Closed loop. No singularities.
                                   = 3 : Closed loop. One singularity.
                                         Not used.
                                   = 4 : Open curve. No singularity.
                                   = 5 : Open curve. Singularity at the
                                         beginning of the curve.
                                   = 6 : Open curve. Singularity at the end
                                         of the curve.
                                   = 7 : Open curve. Singularity at the
                                         beginning  and end of the curve.
                                   = 8 : An isolated singularity. Not used.
                                   = 9 : The curve is exact, pgeom and either
                                         ppar1 or ppar2 is set.      */

  int pretop[4];                 /* Pretopology */
} SISLIntcurve;
```

A.5 The SISLDir Objects

The two SISLdir objects defined in Fortran and the original C structure have the following formats

A.5.1 Fortran C binding object

```
 Type, BIND(C) :: c_SISLdir  ! C direction cone object
    Integer(C_INT) :: igtpi   = 0_C_INT
    Type(C_PTR)    :: ecoef   = C_NULL_PTR
    Real(C_DOUBLE) :: aang    = 0.0_C_DOUBLE
    Type(C_PTR)    :: esmooth = C_NULL_PTR
  End Type
```

A .5.2 Fortran Interface object

```
 Type :: SISLdir  ! Fortran direction cone object
    Integer               :: igtpi      = 0
    Real(REAL64)          :: aang       = 0.0_REAL64
    Real(REAL64), Pointer :: ecoef(:)   => NULL()
    Real(REAL64), Pointer :: esmooth(:) => NULL()
    Type(C_PTR)           :: cptr       = C_NULL_PTR
  End Type
```

A.5.3 SISL C structure (from SISL library) with variable definitions

```
typedef struct SISLdir
{
  int igtpi;                    /* 0 - The direction of the surface or curve
                                      is not greater than pi in any
                                      parameter direction.
                                   1 - The direction of the surface or curve
                                      is greater than pi in the first
                                      parameter direction.
                                   2 - The direction of the surface is greater
                                      than pi in the second parameter
                                      direction.                          */
  double *ecoef;                /* The coordinates to the center of the cone.*/
  double aang;                  /* The angle from the center whice describe the
                                   cone.                                  */
  double *esmooth;              /* Coordinates of object after smoothing.    */
} SISLdir;
```

A.6 The SISLSegmentation Objects

The two SISLsegmentation objects defined in Fortran and the original C structure have the following formats

A.6.1 Fortran C binding object

```
 Type, BIND(C) :: c_SISLSegmentation ! C segmentation object
    Type(C_PTR)    :: seg_val  = C_NULL_PTR
    Type(C_PTR)    :: seg_type = C_NULL_PTR
    Integer(C_INT) :: num_seg  = 0_C_INT
 End Type
```

A.6.2 Fortran Interface object

```
 Type :: SISLSegmentation ! Fortran segmentation object
    Real(REAL64), Pointer  :: seg_val(:) => NULL()
    Integer,      Pointer  :: seg_type(:) => NULL()
    Integer                :: num_seg    = 0
    Type(C_PTR)            :: cptr       = C_NULL_PTR
  End Type
```

A.6.3 SISL C structure (from SISL library) with variable definitions

```
typedef struct SISLSegmentation
{
  double *seg_val; /* Segmentation values */
  int *seg_type;   /* SEGMENTATION_TYPE */
  int num_seg;     /* number of segmentations */
} SISLSegmentation;
```

A.7 The SISLbox Objects

The two SISLbox objects defined in Fortran and the original C structure have the following formats

A.7.1 Fortran C binding object

```
 Type, BIND(C) :: c_SISLbox  ! C bounding box object
    Type(C_PTR)    :: emax      = C_NULL_PTR
    Type(C_PTR)    :: emin      = C_NULL_PTR
    Integer(C_INT) :: imin      = 0_C_INT
    Integer(C_INT) :: imax      = 0_C_INT
    Type(C_PTR)    :: e2max(3)  = [C_NULL_PTR, C_NULL_PTR, C_NULL_PTR]
    Type(C_PTR)    :: e2min(3)  = [C_NULL_PTR, C_NULL_PTR, C_NULL_PTR]
    Real(C_DOUBLE) :: etol(3)   = [0.0_C_DOUBLE, 0.0_C_DOUBLE, 0.0_C_DOUBLE]
  End Type
```

A.7.2 Fortran Interface object

```
 Type  :: SISLbox ! Fortran bounding box object
    Integer               :: imin     = 0
    Integer               :: imax     = 0
    Real(REAL64), Pointer :: emax(:)  => NULL()
    Real(REAL64), Pointer :: emin(:)  => NULL()
    Type(R64_p)           :: e2max(3)
```

```
    Type(R64_p)           :: e2min(3) ! R64_p contains a REAL64 array pointer
    Real(REAL64)          :: etol(3)  = [0.0_REAL64, 0.0_REAL64, 0.0_REAL64]
    Type(C_PTR)           :: cptr     = C_NULL_PTR
  End Type
```

A.7.3 SISL C structure (from SISL library) with variable definitions

```
typedef struct SISLbox
{
  double *emax;                /* The minimum values to the boxes.          */
  double *emin;                /* The maximum values to the boxes.          */
  int imin;                    /* The index of the min coeff (one-dim case) */
  int imax;                    /* The index of the max coeff (one-dim case) */

  double *e2max[3];            /* The minimum values dependant on tolerance.*/
  double *e2min[3];            /* The maximum values dependant on tolerance.*/
  double etol[3];              /* Tolerances of the boxes.                  */
} SISLbox;
```

Appendix B: Listings of the forSISL Fortran Subroutine Interfaces

B.1 Overview

The following listings show just the argument lists for the Fortran routines that make up the forSISL interfaces. For now the user is referred to the SISL Reference Manual (v 4.4) for definitions of the variables and the required dimensions of all arrays passed to the SISL and forSISL routines. These will be added to this Appendix at a later date or moved to a separate document.

B.2 Curve Definition

```
!---------------------------------- s1602 -------------------------------------

  Subroutine s1602(startpt, endpt, order, dim, startpar, endpar, curve, stat)

!! PURPOSE
!!   s1602 - Creates a staight line represented as a B-spline curve between two
!!           points

!! INTERFACE

!!  Subroutine s1602(startpt, endpt, order, dim, startpar, endpar, curve, stat)
!!    Real(REAL64),    Intent(IN)    :: startpt(*)
!!    Real(REAL64),    Intent(IN)    :: endpt(*)
!!    Integer,         Intent(IN)    :: order
!!    Integer,         Intent(IN)    :: dim
!!    Real(REAL64),    Intent(IN)    :: startpar
!!    Real(REAL64),    Intent(INOUT) :: endpar
!!    Type(SISLcurve), Intent(INOUT) :: curve
!!    Integer,         Intent(INOUT) :: stat

!---------------------------------- s1356 -------------------------------------

  Subroutine s1356(epoint, inbpnt, idim, nptyp, icnsta, icnend, iopen, ik,     &
                   astpar, cendpar, rc, gpar, jnbpar, jstat)

!! PURPOSE
!!   s1356 - Compute a curve interpolating a set of points. The points can be
!!           assigned a tangent (derivative). The parameterization of the curve
!!           will be generated and the curve can be open, closed non-periodic
!!           or periodic. If end-conditions are conflicting, the closed curve
!!           rules out other end conditions. The output will be represented as
!!           a B-spline curve.

!! INTERFACE
!!    Subroutine s1356(epoint, inbpnt, idim, nptyp, icnsta, icnend, iopen, ik, &
!!                     astpar, cendpar, rc, gpar, jnbpar, jstat)
!!      Real(REAL64),                    Intent(IN)    :: epoint(*)
!!      Integer,                         Intent(IN)    :: inbpnt
!!      Integer,                         Intent(IN)    :: idim
!!      Integer,                         Intent(IN)    :: nptyp(*)
!!      Integer,                         Intent(IN)    :: icnsta
!!      Integer,                         Intent(IN)    :: icnend
!!      Integer,                         Intent(IN)    :: iopen
```

```
!!      Integer,                        Intent(IN)    :: ik
!!      Real(REAL64),                   Intent(IN)    :: astpar
!!      Real(REAL64),                   Intent(INOUT) :: cendpar
!!      Type(SISLCURVE),                Intent(INOUT) :: rc
!!      Real(REAL64),    ALLOCATABLE,   Intent(INOUT) :: gpar(:)
!!      Integer,                        Intent(INOUT) :: jnbpar
!!      Integer,                        Intent(INOUT) :: jstat

!---------------------------------- s1357 -------------------------------------

  Subroutine s1357(epoint, inbpnt, idim, nptyp, epar, icnsta, icnend, iopen,   &
                   ik, astpar, cendpar, rc, gpar, jnbpar, jstat)

!! PURPOSE
!!   s1357 - Compute a curve interpolating a set of points. The points can be
!!           assigned a tangent (derivative). The curve can be open, closed
!!            or periodic. If end-conditions are conflicting, the closed
!!            curve rules out other end conditions. The parameterization is
!!            given by the array epar. The output will be represented as a
!!            B-spline curve.

!! INTERFACE
!!   Subroutine s1357(epoint, inbpnt, idim, nptyp, epar, icnsta, icnend, iopen,&
!!                    ik, astpar, cendpar, rc, gpar, jnbpar, jstat)
!!     Real(REAL64),                  Intent(IN)    :: epoint(*)
!!     Integer,                       Intent(IN)    :: inbpnt
!!     Integer,                       Intent(IN)    :: idim
!!     Integer,                       Intent(IN)    :: nptyp(*)
!!     Real(REAL64),                  Intent(IN)    :: epar(*)
!!     Integer,                       Intent(IN)    :: icnsta
!!     Integer,                       Intent(IN)    :: icnend
!!     Integer,                       Intent(IN)    :: iopen
!!     Integer,                       Intent(IN)    :: ik
!!     Real(REAL64),                  Intent(IN)    :: astpar
!!     Real(REAL64),                  Intent(INOUT) :: cendpar
!!     Type(SISLCURVE),               Intent(INOUT) :: rc
!!     Real(REAL64),    ALLOCATABLE,  Intent(INOUT) :: gpar(:)
!!     Integer,                       Intent(INOUT) :: jnbpar
!!     Integer,                       Intent(INOUT) :: jstat
!!     Implicit NONE

!---------------------------------- s1380 -------------------------------------

  Subroutine s1380(point, derivate, numpt, dim, typepar, curve, stat)

!! PURPOSE
!!   s1380 - Computes the cubic Hermite interpolant to the data given by the
!!   points point and the derivatives derivate. The output is represented as
!!   a B-spline curve.

!! INTERFACE
!!   Subroutine s1380(point, derivate, numpt, dim, typepar, curve, stat)
!!     Real(REAL64),    Intent(IN)    :: point(*)
!!     Real(REAL64),    Intent(IN)    :: derivate(*)
!!     Integer,         Intent(IN)    :: numpt
!!     Integer,         Intent(IN)    :: dim
!!     Integer,         Intent(IN)    :: typepar
!!     Type(SISLcurve), Intent(INOUT) :: curve
!!     Integer,         Intent(INOUT) :: stat
```

```
!--------------------------------- s1379 -----------------------------------

  Subroutine s1379(point, derivate, par, numpt, dim, curve, stat)

!! PURPOSE
!!   s1379 - Computes the cubic Hermite interpolant to the data given by the
!!           points point and the derivatives derivate and the parameterization
!!           par. The output is represented as a B-spline curve.

!! INTERFACE
!!   Subroutine s1379(point, derivate, par, numpt, dim, curve, stat)
!!     Real(REAL64),    Intent(IN)    :: point(*)
!!     Real(REAL64),    Intent(IN)    :: derivate(*)
!!     Real(REAL64),    Intent(IN)    :: par(*)
!!     Integer,         Intent(IN)    :: numpt
!!     Integer,         Intent(IN)    :: dim
!!     Type(SISLcurve), Intent(INOUT) :: curve
!!     Integer,         Intent(INOUT) :: stat

!--------------------------------- s1607 -----------------------------------

  Subroutine s1607(curve1, curve2, epsge, end1, fillpar1, end2, fillpar2,      &
                   filltype, dim, order, newcurve, stat)

!! PURPOSE
!!   s1607 - Calculates a fillet curve between two curves. The start and end
!!           point for the fillet is given as one parameter value for each of
!!           the curves. The output is represented as a B-spline curve.

!! INTERFACE
!!   Subroutine s1607(curve1, curve2, epsge, end1, fillpar1, end2, fillpar2,   &
!!                    filltype, dim, order, newcurve, stat)
!!     Type(SISLcurve), Intent(IN)    :: curve1
!!     Type(SISLcurve), Intent(IN)    :: curve2
!!     Real(REAL64),    Intent(IN)    :: epsge
!!     Real(REAL64),    Intent(IN)    :: end1
!!     Real(REAL64),    Intent(IN)    :: fillpar1
!!     Real(REAL64),    Intent(IN)    :: end2
!!     Real(REAL64),    Intent(IN)    :: fillpar2
!!     Integer,         Intent(IN)    :: filltype
!!     Integer,         Intent(IN)    :: dim
!!     Integer,         Intent(IN)    :: order
!!     Type(SISLcurve), Intent(INOUT) :: newcurve
!!     Integer,         Intent(INOUT) :: stat

!--------------------------------- s1608 -----------------------------------

  Subroutine s1608(curve1, curve2, epsge, point1, startpt1, point2, endpt2,    &
                   filltype, dim, order, newcurve, parpt1, parspt1, parpt2,   &
                   parept2, stat)

!! PURPOSE
!!   s1608 - Calculates a fillet curve between two curves. Points indicate be-
!!           tween which points on the input curve the fillet is to be produced.
!!           The output is represented as a B-spline curve.

!! INTERFACE
!!   Subroutine s1608(curve1, curve2, epsge, point1, startpt1, point2, endpt2, &
```

```
!!                    filltype, dim, order, newcurve, parpt1, parspt1, parpt2, &
!!                    parept2, stat)
!!     Type(SISLcurve), Intent(IN)    :: curve1
!!     Type(SISLcurve), Intent(IN)    :: curve2
!!     Real(REAL64),    Intent(IN)    :: epsge
!!     Real(REAL64),    Intent(IN)    :: point1(*)
!!     Real(REAL64),    Intent(IN)    :: startpt1(*)
!!     Real(REAL64),    Intent(IN)    :: point2(*)
!!     Real(REAL64),    Intent(IN)    :: endpt2(*)
!!     Integer,         Intent(IN)    :: filltype
!!     Integer,         Intent(IN)    :: dim
!!     Integer,         Intent(IN)    :: order
!!     Type(SISLcurve), Intent(INOUT) :: newcurve
!!     Real(REAL64),    Intent(INOUT) :: parpt1
!!     Real(REAL64),    Intent(INOUT) :: parspt1
!!     Real(REAL64),    Intent(INOUT) :: parpt2
!!     Real(REAL64),    Intent(INOUT) :: parept2
!!     Integer,         Intent(INOUT) :: stat

!---------------------------------- s1609 -------------------------------------

  Subroutine s1609(curve1, curve2, epsge, point1, pointf, point2,  radius,     &
                   normal, filltype, dim, order, newcurve, parend1, parspt1,   &
                   parend2, parept2, stat)
!! PURPOSE
!!   s1609 - Calculates a constant radius fillet curve between two curves if
!!           possible. The output is represented as a B-spline curve.

!! INTERFACE
!!   Subroutine s1609(curve1, curve2, epsge, point1, pointf, point2,  radius,  &
!!                    normal, filltype, dim, order, newcurve, parend1, parspt1, &
!!                    parend2, parept2, stat)
!!     Type(SISLcurve), Intent(IN)    :: curve1
!!     Type(SISLcurve), Intent(IN)    :: curve2
!!     Real(REAL64),    Intent(IN)    :: epsge
!!     Real(REAL64),    Intent(IN)    :: point1(*)
!!     Real(REAL64),    Intent(IN)    :: pointf(*)
!!     Real(REAL64),    Intent(IN)    :: point2(*)
!!     Real(REAL64),    Intent(IN)    :: radius
!!     Real(REAL64),    Intent(IN)    :: normal(*)
!!     Integer,         Intent(IN)    :: filltype
!!     Integer,         Intent(IN)    :: dim
!!     Integer,         Intent(IN)    :: order
!!     Type(SISLcurve), Intent(INOUT) :: newcurve
!!     Real(REAL64),    Intent(INOUT) :: parend1
!!     Real(REAL64),    Intent(INOUT) :: parspt1
!!     Real(REAL64),    Intent(INOUT) :: parend2
!!     Real(REAL64),    Intent(INOUT) :: parept2
!!     Integer,         Intent(INOUT) :: stat

!---------------------------------- s1014 -------------------------------------

  Subroutine s1014(pc1, circ_cen, circ_rad, aepsge, eps1, eps2, aradius,       &
                   parpt1, parpt2, centre, jstat)

!! PURPOSE
!!   s1014 - Computes a circular fillet by iterating to the start and end points
!!           of a fillet between a 2D curve and a circle. The centre of the
!!           circular fillet is also calculated
```

```
!! INTERFACE
!!   Subroutine s1014(pc1, circ_cen, circ_rad, aepsge, eps1, eps2, aradius,    &
!!                    parpt1, parpt2, centre, jstat)
!!     Type(SISLcurve), Intent(IN)    :: pc1
!!     Real(REAL64),    Intent(IN)    :: circ_cen(*)
!!     Real(REAL64),    Intent(IN)    :: circ_rad
!!     Real(REAL64),    Intent(IN)    :: aepsge
!!     Real(REAL64),    Intent(IN)    :: eps1(*)
!!     Real(REAL64),    Intent(IN)    :: eps2(*)
!!     Real(REAL64),    Intent(IN)    :: aradius
!!     Real(REAL64),    Intent(INOUT) :: parpt1
!!     Real(REAL64),    Intent(INOUT) :: parpt2
!!     Real(REAL64),    Intent(INOUT) :: centre(*)
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s1015 -------------------------------------

  Subroutine s1015(pc1, pc2,  aepsge, eps1, eps2, aradius, parpt1, parpt2,     &
                   centre, jstat)

!! PURPOSE
!!   s1015 - Computes a fillet by iterating to the start and end points of a
!!           fillet between two 2D curves. The centre of the circular fillet is
!!           also calculated.

!! INTERFACE
!!   Subroutine s1015(pc1, pc2,  aepsge, eps1, eps2, aradius, parpt1, parpt2, &
!!                    centre, jstat)
!!     Type(SISLcurve), Intent(IN)    :: pc1
!!     Type(SISLcurve), Intent(IN)    :: pc2
!!     Real(REAL64),    Intent(IN)    :: aepsge
!!     Real(REAL64),    Intent(IN)    :: eps1(*)
!!     Real(REAL64),    Intent(IN)    :: eps2(*)
!!     Real(REAL64),    Intent(IN)    :: aradius
!!     Real(REAL64),    Intent(INOUT) :: parpt1
!!     Real(REAL64),    Intent(INOUT) :: parpt2
!!     Real(REAL64),    Intent(INOUT) :: centre(*)
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s1016 -------------------------------------

  Subroutine s1016(pc1, point, normal,  aepsge, eps1, eps2, aradius, parpt1,   &
                   parpt2, centre, jstat)

!! PURPOSE
!!   s1016 - Compute the fillet by iterating to the start and end points of a
!!           fillet between a 2D curve and a 2D line. The centre of the circular
!!           fillet is also calculated

!! INTERFACE
!!   Subroutine s1016(pc1, point, normal,  aepsge, eps1, eps2, aradius, parpt1,&
!!                    parpt2, centre, jstat)
!!     Type(SISLcurve), Intent(IN)    :: pc1
!!     Real(REAL64),    Intent(IN)    :: point(*)
!!     Real(REAL64),    Intent(IN)    :: normal(*)
!!     Real(REAL64),    Intent(IN)    :: aepsge
!!     Real(REAL64),    Intent(IN)    :: eps1(*)
!!     Real(REAL64),    Intent(IN)    :: eps2(*)
```

```
!!     Real(REAL64),    Intent(IN)    :: aradius
!!     Real(REAL64),    Intent(INOUT) :: parpt1
!!     Real(REAL64),    Intent(INOUT) :: parpt2
!!     Real(REAL64),    Intent(INOUT) :: centre(*)
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s1606 -------------------------------------

  Subroutine s1606(curve1, curve2, epsge, point1, point2, blendtype, dim,      &
                   order, newcurve, stat)

!! PURPOSE
!!   s1606 - Computes a blending curve between two curves. Two points
!!           indicate between which ends the blend is to be produced. The
!!           blending curve is either a circle or an approximated conic section
!!           if this is possible, otherwise it is a quadratic polynomial spline
!!           curve. The output is represented as a B-spline curve.

!! INTERFACE
!!   Subroutine s1606(curve1, curve2, epsge, point1, point2, blendtype, dim,   &
!!                    order, newcurve, stat)
!!     Type(SISLcurve), Intent(IN)    :: curve1
!!     Type(SISLcurve), Intent(IN)    :: curve2
!!     Real(REAL64),    Intent(IN)    :: epsge
!!     Real(REAL64),    Intent(IN)    :: point1(*)
!!     Real(REAL64),    Intent(IN)    :: point2(*)
!!     Integer,         Intent(IN)    :: blendtype
!!     Integer,         Intent(IN)    :: dim
!!     Integer,         Intent(IN)    :: order
!!     Type(SISLcurve), Intent(INOUT) :: newcurve
!!     Integer,         Intent(INOUT) :: stat

!---------------------------------- s1303 -------------------------------------

  Subroutine s1303(startpt, epsge, angle, centrept, axis, dim, curve, stat)

!! PURPOSE
!!   s1303 - To create a curve approximating a circular arc around the axis
!!           defined by the centre point, an axis vector, a start point and a
!!           rotational angle. The maximal deviation between the true circular
!!           arc and the approximation to the arc is controlled by the geometric
!!           tolerance (epsge). The output will be represented as a B-spline
!!           curve.

!! INTERFACE
!!   Subroutine s1303(startpt, epsge, angle, centrept, axis, dim, curve, stat)
!!     Real(REAL64),    Intent(IN)    :: startpt(*)
!!     Real(REAL64),    Intent(IN)    :: epsge
!!     Real(REAL64),    Intent(IN)    :: angle
!!     Real(REAL64),    Intent(IN)    :: centrept(*)
!!     Real(REAL64),    Intent(IN)    :: axis(*)
!!     Integer,         Intent(IN)    :: dim
!!     Type(SISLcurve), Intent(INOUT) :: curve
!!     Integer,         Intent(INOUT) :: stat

!---------------------------------- s1611 -------------------------------------

  Subroutine s1611(point, numpt, dim, typept, iopen, order, startpar, epsge,   &
                   endpar, curve, stat)
```

```
!! PURPOSE
!!  s1611 - approximate a conic arc with a curve in two or three dimen-
!!          sional space. If two points are given, a straight line is produced,
!!          if three an approximation of a circular arc, and if four or five a
!!          conic arc. The output will be represented as a B-spline curve.

!! INTERFACE
!!   Subroutine s1611(point, numpt, dim, typept, iopen, order, startpar, epsge,&
!!                    endpar, curve, stat)
!!     Real(REAL64),    Intent(IN)    :: point(*)
!!     Integer,         Intent(IN)    :: numpt
!!     Integer,         Intent(IN)    :: dim
!!     Real(REAL64),    Intent(IN)    :: typept(*)
!!     Integer,         Intent(IN)    :: iopen
!!     Integer,         Intent(IN)    :: order
!!     Real(REAL64),    Intent(IN)    :: startpar
!!     Real(REAL64),    Intent(IN)    :: epsge
!!     Real(REAL64),    Intent(INOUT) :: endpar
!!     Type(SISLcurve), Intent(INOUT) :: curve
!!     Integer,         Intent(INOUT) :: stat

!---------------------------------- s1630 -------------------------------------

  Subroutine s1630(epoint, inbpnt, astpar, iopen, idim, ik, rc, jstat)

!! PURPOSE
!!   s1630 - To compute a curve using the input points as controlling vertices.
!!           The distances between the points are used as parametrization.
!!           The output will be represented as a B-spline curve.

!! INTERFACE
!!   Subroutine s1630(epoint, inbpnt, astpar, iopen, idim, ik, rc, jstat)
!!     Real(REAL64),    Intent(IN)    :: epoint(*)
!!     Integer,         Intent(IN)    :: inbpnt
!!     Real(REAL64),    Intent(IN)    :: astpar
!!     Integer,         Intent(IN)    :: iopen
!!     Integer,         Intent(IN)    :: idim
!!     Integer,         Intent(IN)    :: ik
!!     Type(SISLcurve), Intent(INOUT) :: rc
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s1360 -------------------------------------

  Subroutine s1360(oldcurve, offset, epsge, norm, max, dim, newcurve, stat)

!! PURPOSE
!!   s1360 - To create a approximation of the offset to a curve within a toler-
!!           ance. The output will be represented as a B-spline curve.
!!           With an offset of zero, this routine can be used to approximate any
!!           any NURBS curve, within a tolerance, with a (non-rational) B-spline
!!           curve.

!! INTERFACE
!!   Subroutine s1360(oldcurve, offset, epsge, norm, max, dim, newcurve, stat)
!!     Type(SISLcurve), Intent(IN)    :: oldcurve
!!     Real(REAL64),    Intent(IN)    :: offset
!!     Real(REAL64),    Intent(IN)    :: epsge
!!     Real(REAL64),    Intent(IN)    :: norm(*)
```

```
!!     Real(REAL64),    Intent(IN)    :: max
!!     Integer,         Intent(IN)    :: dim
!!     Type(SISLcurve), Intent(INOUT) :: newcurve
!!     Integer,         Intent(INOUT) :: stat

!---------------------------------- s1613 -------------------------------------

  Subroutine s1613(curve, epsge, points, numpoints, stat)

!! PURPOSE
!!   1613 - To calculate a set of points on a curve. The straight lines between
!!          the points will not deviate more than epsge from the curve at any
!!          point. The generated points will have the same spatial dimension
!!          as the input curve.

!! INTERFACE
!!   Subroutine s1613(curve, epsge, points, numpoints, stat)
!!     Type(SISLcurve),              Intent(IN)    :: curve
!!     Real(REAL64),                 Intent(IN)    :: epsge
!!     Real(REAL64),    ALLOCATABLE, Intent(INOUT) :: points(:)
!!     Integer,                      Intent(INOUT) :: numpoints
!!     Integer,                      Intent(INOUT) :: stat

!---------------------------------- s1600 -------------------------------------

  Subroutine s1600(oldcurve, point, normal, dim, newcurve, stat)

!! PURPOSE
!!   s1600 - To mirror a curve around a plane.

!! INTERFACE
!!   Subroutine s1600(oldcurve, point, normal, dim, newcurve, stat)
!!     Type(SISLcurve), Intent(IN)    :: oldcurve
!!     Real(REAL64),    Intent(IN)    :: point(*)
!!     Real(REAL64),    Intent(IN)    :: normal(*)
!!     Integer,         Intent(IN)    :: dim
!!     Type(SISLcurve), Intent(INOUT) :: newcurve
!!     Integer,         Intent(INOUT) :: stat

!---------------------------------- s1389 -------------------------------------

  Subroutine s1389(curve, cubic, numcubic, dim, stat)

!! PURPOSE
!!   s1389 - Convert a curve of order up to 4 to a sequence of non-rational
!!           cubic segments with uniform parameterization.

!! INTERFACE
!!   Subroutine s1389(curve, cubic, numcubic, dim, stat)
!!     Type(SISLcurve),              Intent(IN)    :: curve
!!     Real(REAL64),    ALLOCATABLE, Intent(INOUT) :: cubic(:)
!!     Integer,                      Intent(INOUT) :: numcubic
!!     Integer,                      Intent(INOUT) :: dim
!!     Integer,                      Intent(INOUT) :: stat

!---------------------------------- s1730 -------------------------------------

  Subroutine s1730(curve, newcurve, stat)
```

```
!! PURPOSE
!!   s1730 - To convert a curve to a sequence of Bezier curves. The Bezier
!!           curves are stored as one curve with all knots of multiplicity
!!           newcurve%ik (order of the curve). If the input curve is rational,
!!           the generated Bezier curves will be rational too (i.e. there will
!!           be rational weights in the representation of the Bezier curves).

!! INTERFACE
!!   Subroutine s1730(curve, newcurve, stat)
!!     Type(SISLcurve), Intent(IN)    :: curve
!!     Type(SISLcurve), Intent(INOUT) :: newcurve
!!     Integer,         Intent(INOUT) :: stat

!---------------------------------- s1732 -----------------------------------

  Subroutine s1732(curve, number, startpar, endpar, coef, stat)

!! PURPOSE
!!   s1732 - To pick out the next Bezier curve from a curve. This function re-
!!           quires a curve represented as the curve that is output from s1730.
!!           If the input curve is rational, the generated Bezier curves will be
!!           rational too (i.e. there will be rational weights in the
!!           representation  of the Bezier curves).

!! INTERFACE
!!   Subroutine s1732(curve, number, startpar, endpar, coef, stat)
!!     Type(SISLcurve),              Intent(IN)    :: curve
!!     Integer,                      Intent(IN)    :: number
!!     Real(REAL64),                 Intent(INOUT) :: startpar
!!     Real(REAL64),                 Intent(INOUT) :: endpar
!!     Real(REAL64),                 Intent(INOUT) :: coef(*)
!!     Integer,                      Intent(INOUT) :: stat

!---------------------------------- s1750 -----------------------------------

  Subroutine s1750(curve, order, newcurve, stat)

!! PURPOSE
!!   s1720 - To express the “i”-th derivative of an open curve as a curve.

!! INTERFACE
!!   Subroutine s1750(curve, order, newcurve, stat)
!!     Type(SISLcurve), Intent(IN)    :: curve
!!     Integer,         Intent(IN)    :: order
!!     Type(SISLcurve), Intent(INOUT) :: newcurve
!!     Integer,         Intent(INOUT) :: stat

!---------------------------------- s1720 -----------------------------------

  Subroutine s1720(curve, derive, newcurve, stat)

!! PURPOSE
!!   s1720 - To express the “i”-th derivative of an open curve as a curve.

!! INTERFACE
!!  Subroutine s1720(curve, derive, newcurve, stat)
!!    Type(SISLcurve), Intent(IN)    :: curve
!!    Integer,         Intent(IN)    :: derive
!!    Type(SISLcurve), Intent(INOUT) :: newcurve
```

```
!!    Integer,         Intent(INOUT) :: stat

!---------------------------------- s1522 -------------------------------------

  Subroutine s1522(normal, centre, ellipaxis, ratio, dim, ellipse, stat)

!! PURPOSE
!!   s1522 - Convert a 2D or 3D analytical ellipse to a curve. The curve will
!!           be geometrically exact.

!! INTERFACE
!!   Subroutine s1522(normal, centre, ellipaxis, ratio, dim, ellipse, stat)
!!     Real(REAL64),    Intent(IN)    :: normal(*)
!!     Real(REAL64),    Intent(IN)    :: centre(*)
!!     Real(REAL64),    Intent(IN)    :: ellipaxis(*)
!!     Real(REAL64),    Intent(IN)    :: ratio
!!     Integer,         Intent(IN)    :: dim
!!     Type(SISLcurve), Intent(INOUT) :: ellipse
!!     Integer,         Intent(INOUT) :: stat

!---------------------------------- s1011 -------------------------------------

  Subroutine s1011(start_pos, top_pos, end_pos, shape, dim, arc_seg, stat)

!! PURPOSE
!!   s1011 - Convert an analytic conic arc to a curve. The curve will be geo-
!!           metrically exact. The arc is given by position at start, shoulder
!!           point and end, and a shape factor.

!! INTERFACE
!!   Subroutine s1011(start_pos, top_pos, end_pos, shape, dim, arc_seg, stat)
!!     Real(REAL64),    Intent(IN)    :: start_pos(*)
!!     Real(REAL64),    Intent(IN)    :: top_pos(*)
!!     Real(REAL64),    Intent(IN)    :: end_pos(*)
!!     Real(REAL64),    Intent(IN)    :: shape
!!     Integer,         Intent(IN)    :: dim
!!     Type(SISLcurve), Intent(INOUT) :: arc_seg
!!     Integer,         Intent(INOUT) :: stat


!---------------------------------- s1012 -------------------------------------

  Subroutine s1012(start_pos, axis_pos, axis_dir, frequency, numb_quad,        &
                   counter_clock, helix, stat)

!! PURPOSE
!!   s1012 - Convert an analytical truncated helix to a curve. The curve will
!!           be geometrically exact.

!! INTERFACE
!!   Subroutine s1012(start_pos, axis_pos, axis_dir, frequency, numb_quad,     &
!!                    counter_clock, helix, stat)
!!     Real(REAL64),    Intent(IN)    :: start_pos(*)
!!     Real(REAL64),    Intent(IN)    :: axis_pos(*)
!!     Real(REAL64),    Intent(IN)    :: axis_dir(*)
!!     Real(REAL64),    Intent(IN)    :: frequency
!!     Integer,         Intent(IN)    :: numb_quad
!!     Integer,         Intent(IN)    :: counter_clock
!!     Type(SISLcurve), Intent(INOUT) :: helix
```

```
!!     Integer,         Intent(INOUT) :: stat
```

B.3 Curve Interrogation

```
!--------------------------------- s1871 -------------------------------------

  Subroutine s1871(pc1, pt1, idim, aepsge, jpt, gpar1, jcrv, wcurve, jstat)

!! PURPOSE
!!   s1871 - Find all the intersections between a curve and a point.

!! INTERFACE
!!   Subroutine s1871(pc1, pt1, idim, aepsge, jpt, gpar1, jcrv, wcurve, jstat)
!!     Type(SISLcurve),                 Intent(IN)    :: pc1
!!     Real(REAL64),                    Intent(IN)    :: pt1(*)
!!     Integer,                         Intent(IN)    :: idim
!!     Real(REAL64),                    Intent(IN)    :: aepsge
!!     Integer,                         Intent(INOUT) :: jpt
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: gpar1(:)
!!     Integer,                         Intent(INOUT) :: jcrv
!!     Type(SISLIntcurve), ALLOCATABLE, Intent(INOUT) :: wcurve(:)
!!     Integer,                         Intent(INOUT) :: jstat

!--------------------------------- s1850 -------------------------------------

  Subroutine s1850(curve, point, normal, dim, epsco, epsge, numintpt,          &
                   intpar, numintcu, intcurve, stat)

!! PURPOSE
!!   s1850 - Find all the intersections between a curve and a plane (if curve
!!           dimension and dim = 3) or a curve and a line (if curve dimension
!!           and dim = 2).

!! INTERFACE
!!   Subroutine s1850(curve, point, normal, dim, epsco, epsge, numintpt,       &
!!                    intpar, numintcu, intcurve, stat)
!!     Type(SISLcurve),                 Intent(IN)    :: curve
!!     Real(REAL64),                    Intent(IN)    :: point(*)
!!     Real(REAL64),                    Intent(IN)    :: normal(*)
!!     Integer,                         Intent(IN)    :: dim
!!     Real(REAL64),                    Intent(IN)    :: epsco
!!     Real(REAL64),                    Intent(IN)    :: epsge
!!     Integer,                         Intent(INOUT) :: numintpt
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: intpar(:)
!!     Integer,                         Intent(INOUT) :: numintcu
!!     Type(SISLIntcurve), ALLOCATABLE, Intent(INOUT) :: intcurve(:)
!!     Integer,                         Intent(INOUT) :: stat

!--------------------------------- s1327 -------------------------------------

  Subroutine s1327(pcold, epoint, enorm1, enorm2, idim, rcnew, jstat)

!! PURPOSE
!!   1327 - Put the equation of the curve pointed at by pcold into two planes
!!          given by the point epoint and the normals enorm1 and enorm2.
!1          The result is an equation where the new two-dimensional curve
!!          rcnew is to be equal to origo.
```

```
!! INTERFACE
!!   Subroutine s1327(pcold, epoint, enorm1, enorm2, idim, rcnew, jstat)
!!     Type(SISLcurve), Intent(IN)    :: pcold
!!     Real(REAL64),    Intent(IN)    :: epoint(*)
!!     Real(REAL64),    Intent(IN)    :: enorm1(*)
!!     Real(REAL64),    Intent(IN)    :: enorm2(*)
!!     Integer,         Intent(IN)    :: idim
!!     Type(SISLcurve), Intent(INOUT) :: rcnew
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s1371 -------------------------------------

  Subroutine s1371(curve, centre, radius, dim, epsco, epsge, numintpt, intpar, &
                   numintcu, intcurve, stat)

!! PURPOSE
!!   s1371 - Find all the intersections between a curve and a sphere (if curve
!!           dimension and dim = 3), or a curve and a circle (if curve dimension
!!           and dim = 2).

!! INTERFACE
!!   Subroutine s1371(curve, centre, radius, dim, epsco, epsge, numintpt,      &
!!                    intpar, numintcu, intcurve, stat)
!!     Type(SISLcurve),                 Intent(IN)    :: curve
!!     Real(REAL64),                    Intent(IN)    :: centre(*)
!!     Real(REAL64),                    Intent(IN)    :: radius
!!     Integer,                         Intent(IN)    :: dim
!!     Real(REAL64),                    Intent(IN)    :: epsco
!!     Real(REAL64),                    Intent(IN)    :: epsge
!!     Integer,                         Intent(INOUT) :: numintpt
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: intpar(:)
!!     Integer,                         Intent(INOUT) :: numintcu
!!     Type(SISLIntcurve), ALLOCATABLE, Intent(INOUT) :: intcurve(:)
!!     Integer,                         Intent(INOUT) :: stat

!---------------------------------- s1374 -------------------------------------

  Subroutine s1374(curve, conarray, dim, epsco, epsge, numintpt, intpar,      &
                   numintcu, intcurve, stat)

!! PURPOSE
!!   s1374 - Find all the intersections between a curve and a quadric curve, (if
!!           curve dimension and dim = 2), or a curve and a quadric surface,
!!           (if curve dimension and dim = 3).

!! INTERFACE
!!   Subroutine s1374(curve, conarray, dim, epsco, epsge, numintpt, intpar,    &
!!                    numintcu, intcurve, stat)
!!     Type(SISLcurve),                 Intent(IN)    :: curve
!!     Real(REAL64),                    Intent(IN)    :: conarray(*)
!!     Integer,                         Intent(IN)    :: dim
!!     Real(REAL64),                    Intent(IN)    :: epsco
!!     Real(REAL64),                    Intent(IN)    :: epsge
!!     Integer,                         Intent(INOUT) :: numintpt
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: intpar(:)
!!     Integer,                         Intent(INOUT) :: numintcu
!!     Type(SISLIntcurve), ALLOCATABLE, Intent(INOUT) :: intcurve(:)
!!     Integer,                         Intent(INOUT) :: stat
```

```
!--------------------------------- s1857 -----------------------------------

  Subroutine s1857(curve1, curve2, epsco, epsge, numintpt, intpar1, intpar2,   &
                   numintcu, intcurve, stat)

!! PURPOSE
!!   s1857 - Find all the intersections between two curves.

!! INTERFACE
!!   Subroutine s1857(curve1, curve2, epsco, epsge, numintpt, intpar1, intpar2,   &
!!                    numintcu, intcurve, stat)
!!     Type(SISLcurve),                 Intent(IN)    :: curve1
!!     Type(SISLcurve),                 Intent(IN)    :: curve2
!!     Real(REAL64),                    Intent(IN)    :: epsco
!!     Real(REAL64),                    Intent(IN)    :: epsge
!!     Integer,                         Intent(INOUT) :: numintpt
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: intpar1(:)
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: intpar2(:)
!!     Integer,                         Intent(INOUT) :: numintcu
!!     Type(SISLIntcurve), ALLOCATABLE, Intent(INOUT) :: intcurve(:)
!!     Integer,                         Intent(INOUT) :: stat

!--------------------------------- s1240 -----------------------------------

  Subroutine s1240(curve, epsge, length, stat)

!! PURPOSE
!!   s1240 - Compute the length of a curve. The length calculated will not
!!           deviate more than epsge divided by the calculated length, from
!!           the real length of the curve.

!! INTERFACE
!!   Subroutine s1240(curve, epsge, length, stat)
!!     Type(SISLcurve), Intent(IN)    :: curve
!!     Real(REAL64),    Intent(IN)    :: epsge
!!     Real(REAL64),    Intent(INOUT) :: length
!!     Integer,         Intent(INOUT) :: stat

!--------------------------------- s1364 -----------------------------------

  Subroutine s1364(curve, epsge, stat)

!! PURPOSE
!!   s1364 - To check if a curve is closed, i.e. test if the distance between
!!           the end points of the curve is less than a given tolerance.

!! INTERFACE
!!   Subroutine s1364(curve, epsge, stat)
!!     Type(SISLcurve), Intent(IN)    :: curve
!!     Real(REAL64),    Intent(IN)    :: epsge
!!     Integer,         Intent(INOUT) :: stat

!--------------------------------- s1451 -----------------------------------

  Subroutine s1451(pc1, aepsge, jdgen, jstat)

!! PURPOSE
!!   s1451 - To check if a curve is degenerated.
```

```
!! INTERFACE
!!   Subroutine s1451(pc1, aepsge, jdgen, jstat)
!!     Type(SISLcurve), Intent(IN)    :: pc1
!!     Real(REAL64),    Intent(IN)    :: aepsge
!!     Integer,         Intent(INOUT) :: jdgen
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s1363 -------------------------------------

  Subroutine s1363(curve, startpar, endpar, stat)

!! PURPOSE
!!   s1363 - To pick the parameter range of a curve.

!! INTERFACE
!!   Subroutine s1363(curve, startpar, endpar, stat)
!!     Type(SISLcurve), Intent(IN)    :: curve
!!     Real(REAL64),    Intent(INOUT) :: startpar
!!     Real(REAL64),    Intent(INOUT) :: endpar
!!     Integer,         Intent(INOUT) :: stat

!---------------------------------- s1953 -------------------------------------

  Subroutine s1953(curve, point, dim, epsco, epsge, numintpt, intpar,         &
                   numintcu, intcurve, stat)

!! PURPOSE
!!   s1953 - Find the closest points between a curve and a point.

!! INTERFACE
!!   Subroutine s1953(curve, point, dim, epsco, epsge, numintpt, intpar,       &
!!                    numintcu, intcurve, stat)
!!     Type(SISLcurve),                 Intent(IN)    :: curve
!!     Real(REAL64),                    Intent(IN)    :: point(*)
!!     Integer,                         Intent(IN)    :: dim
!!     Real(REAL64),                    Intent(IN)    :: epsco
!!     Real(REAL64),                    Intent(IN)    :: epsge
!!     Integer,                         Intent(INOUT) :: numintpt
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: intpar(:)
!!     Integer,                         Intent(INOUT) :: numintcu
!!     Type(SISLIntcurve), ALLOCATABLE, Intent(INOUT) :: intcurve(:)
!!     Integer,                         Intent(INOUT) :: stat

!---------------------------------- s1957 -------------------------------------

  Subroutine s1957(pcurve, epoint, idim, aepsco, aepsge, gpar, dist, jstat)

!! PURPOSE
!!   s1957 - Find the closest point between a curve and a point. The method is
!!           fast and should work well in clear cut cases but does not guarantee
!!           finding the right solution. As long as it doesn't fail, it will
!!           find exactly one point. In other cases, use s1953().

!! INTERFACE
!!   Subroutine s1957(pcurve, epoint, idim, aepsco, aepsge, gpar, dist, jstat)
!!     Type(SISLcurve), Intent(IN)    :: pcurve
!!     Real(REAL64),    Intent(IN)    :: epoint(*)
!!     Integer,         Intent(IN)    :: idim
!!     Real(REAL64),    Intent(IN)    :: aepsco
```

```
!!     Real(REAL64),    Intent(IN)    :: aepsge
!!     Real(REAL64),    Intent(INOUT) :: gpar
!!     Real(REAL64),    Intent(INOUT) :: dist
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s1774 -------------------------------------

  Subroutine s1774(crv, point, dim, epsge, start, end, guess, clpar, stat)

!! PURPOSE
!!   s1774 - Newton iteration on the distance function between a curve and a
!!           point, to find a closest point or an intersection point. If a bad
!!           choice for the guess parameter is given in, the iteration may end
!!           at a local, not global closest point.

!! INTERFACE
!!   Subroutine s1774(crv, point, dim, epsge, start, end, guess, clpar, stat)
!!     Type(SISLcurve), Intent(IN)    :: crv
!!     Real(REAL64),    Intent(IN)    :: point(*)
!!     Integer,         Intent(IN)    :: dim
!!     Real(REAL64),    Intent(IN)    :: epsge
!!     Real(REAL64),    Intent(IN)    :: start
!!     Real(REAL64),    Intent(IN)    :: end
!!     Real(REAL64),    Intent(IN)    :: guess
!!     Real(REAL64),    Intent(INOUT) :: clpar
!!     Integer,         Intent(INOUT) :: stat

!---------------------------------- s1955 -------------------------------------

  Subroutine s1955(curve1, curve2, epsco, epsge, numintpt, intpar1, intpar2,   &
                   numintcu, intcurve, stat)

!! PURPOSE
!!   s1955 - Find the closest points between two curves.

!! INTERFACE
!!   Subroutine s1955(curve1, curve2, epsco, epsge, numintpt, intpar1,         &
!!                    intpar2, numintcu, intcurve, stat)
!!     Type(SISLcurve),                Intent(IN)    :: curve1
!!     Type(SISLcurve),                Intent(IN)    :: curve2
!!     Real(REAL64),                   Intent(IN)    :: epsco
!!     Real(REAL64),                   Intent(IN)    :: epsge
!!     Integer,                        Intent(INOUT) :: numintpt
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: intpar1(:)
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: intpar2(:)
!!     Integer,                        Intent(INOUT) :: numintcu
!!     Type(SISLIntcurve), ALLOCATABLE, Intent(INOUT) :: intcurve(:)
!!     Integer,                        Intent(INOUT) :: stat

!---------------------------------- s1013 -------------------------------------

  Subroutine s1013(pcurve, ang, ang_tol, guess_par, iter_par, jstat)

!! PURPOSE
!!   s1013 - Find a point on a 2D curve along a given direction.

!! INTERFACE
!!   Subroutine s1013(pcurve, ang, ang_tol, guess_par, iter_par, jstat)
!!     Type(SISLcurve), Intent(IN)    :: pcurve
```

```
!!     Real(REAL64),    Intent(IN)    :: ang
!!     Real(REAL64),    Intent(IN)    :: ang_tol
!!     Real(REAL64),    Intent(IN)    :: guess_par
!!     Real(REAL64),    Intent(INOUT) :: iter_par
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s1920 -------------------------------------

  Subroutine s1920(curve, dir, dim, epsco, epsge, numintpt, intpar,         &
                   numintcu, intcurve, stat)

!! PURPOSE
!!   s1920 - Find the absolute extremal points/intervals of a curve relative to
!!           a given direction.

!! INTERFACE
!!   Subroutine s1920(curve, dir, dim, epsco, epsge, numintpt, intpar,        &
!!                    numintcu, intcurve, stat)
!!     Type(SISLcurve),                 Intent(IN)    :: curve
!!     Real(REAL64),                    Intent(IN)    :: dir(*)
!!     Integer,                         Intent(IN)    :: dim
!!     Real(REAL64),                    Intent(IN)    :: epsco
!!     Real(REAL64),                    Intent(IN)    :: epsge
!!     Integer,                         Intent(INOUT) :: numintpt
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: intpar(:)
!!     Integer,                         Intent(INOUT) :: numintcu
!!     Type(SISLIntcurve), ALLOCATABLE, Intent(INOUT) :: intcurve(:)
!!     Integer,                         Intent(INOUT) :: stat

!---------------------------------- s1241 -------------------------------------

  Subroutine s1241(pcurve, point, dim, epsge, area, stat)

!! PURPOSE
!!   s1241 - To calculate the area between a 2D curve and a 2D point. When
!!           the curve is rotating counter-clockwise around the point, the area
!!           contribution is positive. When the curve is rotating clockwise
!!           around the point, the area contribution is negative. If the curve
!!           is closed or periodic, the area calculated is independent of where
!!           the point is situated. The area is calculated exactly for B-spline
!!           curves, for NURBS the result is an approximation. This routine
!!           will only perform if the order of the curve is less than 7 (can
!!           easily be extended).

!! INTERFACE
!!   Subroutine s1241(pcurve, point, dim, epsge, area, stat)
!!     Type(SISLcurve), Intent(IN)    :: pcurve
!!     Real(REAL64),    Intent(IN)    :: point(*)
!!     Integer,         Intent(IN)    :: dim
!!     Real(REAL64),    Intent(IN)    :: epsge
!!     Real(REAL64),    Intent(INOUT) :: area
!!     Integer,         Intent(INOUT) :: stat

!---------------------------------- s1243 -------------------------------------

  Subroutine s1243(pcurve, point, dim, epsge, weight, area, moment, stat)

!! PURPOSE
!!   s1243 - To calculate the weight point and rotational momentum of an area
```

```
!!           between a 2D curve and a 2D point. The area is also calculated.
!!           When the curve is rotating counter-clockwise around the point, the
!!           area contribution is positive. When the curve is rotating clockwise
!!           around the point, the area contribution is negative. OBSERVE:
!!           FOR CALCULATION OF AREA ONLY, USE s1241().

!! INTERFACE
!!   Subroutine s1243(pcurve, point, dim, epsge, weight, area, moment, stat)
!!     Type(SISLcurve), Intent(IN)    :: pcurve
!!     Real(REAL64),    Intent(IN)    :: point(*)
!!     Integer,         Intent(IN)    :: dim
!!     Real(REAL64),    Intent(IN)    :: epsge
!!     Real(REAL64),    Intent(INOUT) :: weight(*)
!!     Real(REAL64),    Intent(INOUT) :: area
!!     Real(REAL64),    Intent(INOUT) :: moment
!!     Integer,         Intent(INOUT) :: stat

!---------------------------------- newbox -------------------------------------

  Subroutine newbox(idim, box)

!! PURPOSE
!!   Create and initialize a curve/surface bounding box instance

!! INTERFACE
!!   Subroutine newbox(idim, box)
!!     Integer,       Intent(INOUT) :: idim
!!     Type(SISLbox), Intent(INOUT) :: box

!---------------------------------- s1988 --------------------------------------

  Subroutine s1988(pc, emax, emin, jstat)

!! PURPOSE
!!   s1988 - Find the bounding box of a SISLCurve. NB. The geometric
!!           bounding box is returned also in the rational case, that is the
!!           box in homogenous coordinates is NOT computed

!! INTERFACE
!!   Subroutine s1988(pc, emax, emin, jstat)
!!     Type(SISLcurve),                Intent(IN)    :: pc
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: emax(:)
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: emin(:)
!!     Integer,                         Intent(INOUT) :: jstat

!---------------------------------- newdir -------------------------------------

  Subroutine newdir(idim, dir)

!! PURPOSE
!!   newdir - Create and initialize a curve/surface direction instance

!! INTERFACE
!!   Subroutine newdir(idim, dir)
!!     Integer,       Intent(IN)    :: idim
!!     Type(SISLdir), Intent(INOUT) :: dir

!---------------------------------- s1986 --------------------------------------
```

```
  Subroutine s1986(pc, aepsge, jgtpi, gaxis, cang, jstat)

!! PURPOSE
!!   s1986 - Find the direction cone of a curve.

!! INTERFACE
!!   Subroutine s1986(pc, aepsge, jgtpi, gaxis, cang, jstat)
!!     Type(SISLcurve),                Intent(IN)    :: pc
!!     Real(REAL64),                   Intent(IN)    :: aepsge
!!     Integer,                        Intent(INOUT) :: jgtpi
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: gaxis(:)
!!     Real(REAL64),                   Intent(INOUT) :: cang
!!     Integer,                        Intent(INOUT) :: jstat
```

B.4 Curve Analysis

```
!---------------------------------- s2550 -------------------------------------

  Subroutine s2550(curve, ax, num_ax, curvature, jstat)

!! PURPOSE
!!   s2550 - Evaluate the curvature of a curve at given parameter values ax( 1
!!           ),...,ax(num_ax ).

!! INTERFACE
!!   Subroutine s2550(curve, ax, num_ax, curvature, jstat)
!!     Type(SISLcurve), Intent(IN)    :: curve
!!     Real(REAL64),    Intent(IN)    :: ax(*)
!!     Integer,         Intent(IN)    :: num_ax
!!     Real(REAL64),    Intent(INOUT) :: curvature(*)
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s2553 -------------------------------------

  Subroutine s2553(curve, ax, num_ax, torsion, jstat)

!! PURPOSE
!!   s2553 - Evaluate the torsion of a curve at given parameter values ax( 1
!!           ),...,ax( num_ax ).

!! INTERFACE
!!   Subroutine s2553(curve, ax, num_ax, torsion, jstat)
!!     Type(SISLcurve), Intent(IN)    :: curve
!!     Real(REAL64),    Intent(IN)    :: ax(*)
!!     Integer,         Intent(IN)    :: num_ax
!!     Real(REAL64),    Intent(INOUT) :: torsion(*)
!!     Integer,         Intent(INOUT) :: jstat


!---------------------------------- s2556 -------------------------------------

  Subroutine s2556(curve, ax, num_ax, VoC, jstat)

!! PURPOSE
!!   s2556 - Evaluate the Variation of Curvature (VoC) of a curve at given
!!           parameter values ax( 1 ),...,ax( num_ax ).
```

```
!! INTERFACE
!!   Subroutine s2556(curve, ax, num_ax, VoC, jstat)
!!     Type(SISLcurve), Intent(IN)    :: curve
!!     Real(REAL64),    Intent(IN)    :: ax(*)
!!     Integer,         Intent(IN)    :: num_ax
!!     Real(REAL64),    Intent(INOUT) :: VoC(*)
!!     Integer,         Intent(INOUT) :: jstat


!---------------------------------- s2559 -------------------------------------

  Subroutine s2559(curve, ax, num_ax, p, t, n, b, jstat)

!! PURPOSE
!!   s2559 - Evaluate the Frenet Frame (t,n,b) of a curve at given parameter
!!           values ax( 1 ),...,ax( num_ax).

!! INTERFACE
!!   Subroutine s2559(curve, ax, num_ax, p, t, n, b, jstat)
!!     Type(SISLcurve), Intent(IN)    :: curve
!!     Real(REAL64),    Intent(IN)    :: ax(*)
!!     Integer,         Intent(IN)    :: num_ax
!!     Real(REAL64),    Intent(INOUT) :: p(*)
!!     Real(REAL64),    Intent(INOUT) :: t(*)
!!     Real(REAL64),    Intent(INOUT) :: n(*)
!!     Real(REAL64),    Intent(INOUT) :: b(*)
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s2562 -------------------------------------

  Subroutine s2562(curve, ax, num_ax, val_flag, p, t, n, b, val, jstat)

!! PURPOSE
!!   s2562 - Evaluate the 3D position, the Frenet Frame (t,n,b) and geometric
!!           property (curvature, torsion or variation of curvature) of a curve
!!           at given parameter values ax(1),...,ax(num_ ax). These data are
!!           needed to produce spike plots (using the Frenet Frame and the
!!           geometric property) and circular tube plots (using circular in the
!!           normal plane (t,b), where the radius is equal to the geometric
!!           property times a scaling factor for visual effects).

!! INTERFACE
!!   Subroutine s2562(curve, ax, num_ax, val_flag, p, t, n, b, val, jstat)
!!     Type(SISLcurve), Intent(IN)    :: curve
!!     Real(REAL64),    Intent(IN)    :: ax(*)
!!     Integer,         Intent(IN)    :: num_ax
!!     Integer,         Intent(IN)    :: val_flag
!!     Real(REAL64),    Intent(INOUT) :: p(*)
!!     Real(REAL64),    Intent(INOUT) :: t(*)
!!     Real(REAL64),    Intent(INOUT) :: n(*)
!!     Real(REAL64),    Intent(INOUT) :: b(*)
!!     Real(REAL64),    Intent(INOUT) :: val(*)
!!     Integer,         Intent(INOUT) :: jstat
```

B.5 Curve Utilities

```
!------------------------------- newCurve ----------------------------------

  Subroutine newCurve(number, order, knots, coef, ikind, idim, copy, curve)

!! PURPOSE
!!   newCurve - Create and intialize a SISLCurve-instance

!! INTERFACE
!!   Subroutine newCurve(number, order, knots, coef, ikind, idim, copy, curve)
!!     Integer,                Intent(IN)    :: number
!!     Integer,                Intent(IN)    :: order
!!     Real(REAL64),           Intent(IN)    :: knots(*)
!!     Real(REAL64),           Intent(IN)    :: coef(*)
!!     Integer,                Intent(IN)    :: ikind
!!     Integer,                Intent(IN)    :: idim
!!     Integer,                Intent(IN)    :: copy
!!    Type(SISLcurve), TARGET, Intent(INOUT) :: curve


!------------------------------- copyCurve ---------------------------------

  Subroutine copyCurve(old_curve, new_curve)

!! PURPOSE
!!   Make a copy of a curve

!! INTERFACE
!!   Subroutine copyCurve(old_curve, new_curve)
!!     Type(SISLCurve), TARGET, Intent(IN)    :: new_curve
!!     Type(SISLCurve), TARGET, Intent(INOUT) :: new_curve

!------------------------------- freeCurve ---------------------------------

  Subroutine freeCurve(curve, free_cptr)

!! PURPOSE
!!   freeCurve - Free the space occupied by the curve. Before using freeCurve,
!!               make sure the curve object exists.

!! INTERFACE
!!   Subroutine freeCurve(curve, free_cptr)
!!     Type(SISLcurve), TARGET,   Intent(INOUT) :: curve
!!     Logical,         OPTIONAL, Intent(IN)    :: free_cptr

!------------------------------- s1227 -------------------------------------

  Subroutine s1227(curve, der, parvalue, leftknot, deriv, stat)

!! PURPOSE
!!   s1227 - To compute the position and the first derivatives of the curve at
!!           a given parameter value. Evaluation from the left hand side.

!! INTERFACE
!!   Subroutine s1227(curve, der, parvalue, leftknot, deriv, stat)
!!     Type(SISLcurve), Intent(IN)    :: curve
!!     Integer,         Intent(IN)    :: der
!!     Real(REAL64),    Intent(IN)    :: parvalue
```

```
!!     Integer,         Intent(INOUT) :: leftknot
!!     Real(REAL64),    Intent(INOUT) :: deriv(*)
!!     Integer,         Intent(INOUT) :: stat

!---------------------------------- s1221 -------------------------------------

  Subroutine s1221(curve, der, parvalue, leftknot, deriv, stat)

!! PURPOSE
!!   s1221 - To compute the positione and the first derivatives of a curve at a
!!           given parameter value. Evaluation from the right hand side.

!! INTERFACE
!!   Subroutine s1221(curve, der, parvalue, leftknot, deriv, stat)
!!     Type(SISLcurve), Intent(IN)    :: curve
!!     Integer,         Intent(IN)    :: der
!!     Real(REAL64),    Intent(IN)    :: parvalue
!!     Integer,         Intent(INOUT) :: leftknot
!!     Real(REAL64),    Intent(INOUT) :: deriv(*)
!!     Integer,         Intent(INOUT) :: stat

!---------------------------------- s1225 -------------------------------------

  Subroutine s1225(curve, der, parvalue, leftknot, deriv, curvature,           &
                   radius_of_curvature, jstat)

!! PURPOSE
!!   s1225 - Evaluate position, first derivative, curvature and radius of curva-
!!           ture of a curve at a given parameter value, from the left hand
!!           side.

!! INTERFACE
!!   Subroutine s1225(curve, der, parvalue, leftknot, deriv, curvature,        &
!!                    radius_of_curvature, jstat)
!!     Type(SISLcurve), Intent(IN)    :: curve
!!     Integer,         Intent(IN)    :: der
!!     Real(REAL64),    Intent(IN)    :: parvalue
!!     Integer,         Intent(INOUT) :: leftknot
!!     Real(REAL64),    Intent(INOUT) :: deriv(*)
!!     Real(REAL64),    Intent(INOUT) :: curvature(*)
!!     Real(REAL64),    Intent(INOUT) :: radius_of_curvature
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s1226 -------------------------------------

  Subroutine s1226(curve, der, parvalue, leftknot, deriv, curvature,           &
                   radius_of_curvature, jstat)

!! PURPOSE
!!   s1226 - Evaluate position, first derivative, curvature and radius of curva-
!!           ture of a curve at a given parameter value, from the right hand
!!           side.

!! INTERFACE
!!   Subroutine s1226(curve, der, parvalue, leftknot, deriv, curvature,        &
!!                    radius_of_curvature, jstat)
!!     Type(SISLcurve), Intent(IN)    :: curve
!!     Integer,         Intent(IN)    :: der
!!     Real(REAL64),    Intent(IN)    :: parvalue
```

```
!!     Integer,         Intent(INOUT) :: leftknot
!!     Real(REAL64),    Intent(INOUT) :: deriv(*)
!!     Real(REAL64),    Intent(INOUT) :: curvature(*)
!!     Real(REAL64),    Intent(INOUT) :: radius_of_curvature
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s1542 -------------------------------------

  Subroutine s1542(pc1, m, x, eder, jstat)

!! PURPOSE
!!   s1542 - Evaluate the curve defined by pc1 over a m grid of points
!!           (x(i)). Only positions are evaluated. This does not work for in the
!!                   rational case.

!! INTERFACE
!!   Subroutine s1542(pc1, m, x, eder, jstat)
!!     Type(SISLcurve), Intent(IN)    :: pc1
!!     Integer,         Intent(IN)    :: m
!!     Real(REAL64),    Intent(INOUT) :: x(*)
!!     Real(REAL64),    Intent(INOUT) :: eder(*)
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s1710 -------------------------------------

  Subroutine s1710(pc1, apar, rcnew1, rcnew2, jstat)

!! PURPOSE
!!   s1710 - Subdivide a curve at a given parameter value.
!!           NOTE: When the curve is periodic (i.e. when the cuopen flag of
!!           the curve has value = −1), this function will return only ONE
!!           curve through rcnew1. This curve is the same geometric curve as
!!           pc1, but is represented on a closed basis, i.e. with k-tuple start/end
!!           the knots and coinciding start/end coefficients. The cuopen flag of
!!           curve will then be set to closed (= 0) and a status value jstat
!!           equal to 2 will be returned.

!! INTERFACE
!!   Subroutine s1710(pc1, apar, rcnew1, rcnew2, jstat)
!!     Type(SISLcurve), Intent(IN)    :: pc1
!!     Real(REAL64),    Intent(IN)    :: apar
!!     Type(SISLcurve), Intent(INOUT) :: rcnew1
!!     Type(SISLcurve), Intent(INOUT) :: rcnew2
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s1017 -------------------------------------

  Subroutine s1017(pc, rc, apar, jstat)

!!PURPOSE
!!  s1017 - Insert a given knot into the description of a curve.
!!          NOTE : When the curve is periodic (i.e. the curve flag cuopen =
!!          −1), the input parameter value must lie in the half-open [et(kk −
!!         1), el(kn) interval, the function will automatically update the extra
!!          knots and coeffisients. rcnew->in is still equal to pc->in + 1!

!! INTERFACE
!!   Subroutine s1017(pc, rc, apar, jstat)
!!     Type(SISLcurve), Intent(IN)    :: pc
```

```
!!     Type(SISLcurve), Intent(INOUT) :: rc
!!     Real(REAL64),    Intent(IN)    :: apar
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s1018 -------------------------------------

  Subroutine s1018(pc, epar, inpar, rcnew, jstat)

!! PURPOSE
!!   s1018 - Insert a given set of knots into the description of a curve.
!!           NOTE : When the curve is periodic (i.e. when the curve flag
!!           cuopen = −1), the input parameter values must lie in the half-
!!           open (et(kk), et(kn+1), the function will automatically update
!!           the extra knots and coeffisients. The rcnew%in will still be equal
!!           to pc%in + inpar.

!! INTERFACE
!!   Subroutine s1018(pc, epar, inpar, rcnew, jstat)
!!     Type(SISLcurve), Intent(IN)    :: pc
!!     Real(REAL64),    Intent(IN)    :: epar(*)
!!     Integer,         Intent(IN)    :: inpar
!!     Type(SISLcurve), Intent(INOUT) :: rcnew
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s1714 -------------------------------------

  Subroutine s1714(curve, parval1, parval2, newcurve1, newcurve2, stat)

!! PURPOSE
!!   s1714 - Split a curve in two parts at two specified parameter values. The
!!           first curve starts at parval1. If the curve is open, the last part
!!           of the curve is translated so that the end of the curve joins the
!!           start.

!! INTERFACE
!!   Subroutine s1712(curve, parval1, parval2, newcurve1, newcurve2, stat)
!!     Type(SISLcurve), Intent(IN)    :: curve
!!     Real(REAL64),    Intent(IN)    :: parval1
!!     Real(REAL64),    Intent(IN)    :: parval2
!!     Type(SISLcurve), Intent(INOUT) :: newcurve1
!!     Type(SISLcurve), Intent(INOUT) :: newcurve2
!!     Integer,         Intent(INOUT) :: stat

!---------------------------------- s1712 -------------------------------------

  Subroutine s1712(curve, begpar, endpar, newcurve, stat)

!! PURPOSE
!!   s1712 - To pick one part of a curve and make a new curve of the part. If
!!           endpar < begpar the direction of the new curve is turned. Use
!!           s1713() to pick a curve part crossing the start/end points of a
!!           closed (or periodic) curve.

!! INTERFACE
!!   Subroutine s1712(curve, begpar, endpar, newcurve, stat)
!!     Type(SISLcurve), Intent(IN)    :: curve
!!     Real(REAL64),    Intent(IN)    :: begpar
!!     Real(REAL64),    Intent(IN)    :: endpar
!!     Type(SISLcurve), Intent(INOUT) :: newcurve
```

```
!!     Integer,         Intent(INOUT) :: stat

!---------------------------------- s1713 -------------------------------------

  Subroutine s1713(curve, begpar, endpar, newcurve, stat)

!! PURPOSE
!!   s1713 - To pick one part of a closed curve and make a new curve of that
!!           part. If the routine is used on an open curve and endpar ≤ begpar,
!!           the last part of the curve is translated so that the end of the
!!           curve joins the start.

!! INTERFACE
!!   Subroutine s1713(curve, begpar, endpar, newcurve, stat)
!!     Type(SISLcurve), Intent(IN)    :: curve
!!     Real(REAL64),    Intent(IN)    :: begpar
!!     Real(REAL64),    Intent(IN)    :: endpar
!!     Type(SISLcurve), Intent(INOUT) :: newcurve
!!     Integer,         Intent(INOUT) :: stat

!---------------------------------- s1715 -------------------------------------

  Subroutine s1715(curve1, curve2, end1, end2, newcurve, stat)

!! PURPOSE
!!   s1715 - To join one end of one curve with one end of another curve by
!!           translating the second curve. If curve1 is to be joined at the
!!           start, the direction of the curve is turned. If curve2 is to be
!!           joined at the end, the direction of this curve is turned. This
!!           means that curve1 always makes the first part of the new curve.

!! INTERFACE
!!   Subroutine s1715(curve1, curve2, end1, end2, newcurve, stat)
!!     Type(SISLcurve), Intent(IN)    :: curve1
!!     Type(SISLcurve), Intent(IN)    :: curve2
!!     Integer,         Intent(IN)    :: end1
!!     Integer,         Intent(IN)    :: end2
!!     Type(SISLcurve), Intent(INOUT) :: newcurve
!!     Integer,         Intent(INOUT) :: stat


!---------------------------------- s1716 -------------------------------------

  Subroutine s1716(curve1, curve2, espge, newcurve, stat)

!! PURPOSE
!!   s1716 - To join two curves at the ends that lie closest to each other, if
!!           the distance between the ends is less than the tolerance epsge. If
!!           curve1 is to be joined at the start, the direction of the curve is
!!           turned. If curve2 is to be joined at the end, the direction of this
!!           curve is turned. This means that curve1 always makes up the first
!!           part of the new curve. If epsge is positive, but smaller than the
!!           smallest distance between the ends of the two curves, a NULL
!!           pointer is returned.

!! INTERFACE
!!   Subroutine s1716(curve1, curve2, espge, newcurve, stat)
!!     Type(SISLcurve), Intent(IN)    :: curve1
!!     Type(SISLcurve), Intent(IN)    :: curve2
```

```
!!     Real(REAL64),    Intent(IN)    :: espge
!!     Type(SISLcurve), Intent(INOUT) :: newcurve
!!     Integer,         Intent(INOUT) :: stat

!---------------------------------- s1706 -------------------------------------

  Subroutine s1706(curve)

!! PURPOSE
!!   s1706 - Turn the direction of a curve by reversing the ordering of the
!!           coefficients. The start parameter value of the new curve is the
!!           same as the start parameter value of the old curve. This routine
!!           turns the direction of the orginal curve. If you want a copy with
!!           a turned direction, just make a copy and turn the direction of the
!!           copy.

!! INTERFACE
!!   Subroutine s1706(curve)
!!     Type(SISLcurve), Intent(INOUT) :: curve

!---------------------------------- s1233 -------------------------------------

  Subroutine s1233(pc, afak1, afak2, rc, jstat)

!! PURPOSE
!!   s1233 - To extend a B-spline curve (i.e. NOT rationals) at the start and/or
!!           the end of the curve by continuing the polynomial behaviour of
!!           the curve.

!! INTERFACE
!!   Subroutine s1233(pc, afak1, afak2, rc, jstat)
!!     Type(SISLCurve), Intent(IN)    :: pc
!!     Real(REAL64),    Intent(IN)    :: afak1
!!     Real(REAL64),    Intent(IN)    :: afak2
!!     Type(SISLCurve), Intent(INOUT) :: rc
!!     Integer,         Intent(INOUT) :: jstat
```

B.6 Surface Definition

```
!---------------------------------- s1536 -------------------------------------

  Subroutine s1536(points, im1, im2, idim, ipar, con1, con2, con3, con4,       &
                   iorder1, iorder2, iopen1, iopen2, rsurf, jstat)

!! PURPOSE
!!   s1536 - To compute a tensor surface interpolating a set of points, auto-
!!           matic parameterization. The output is represented as a B-spline
!!           surface.

!! INTERFACE
!!   Subroutine s1536(points, im1, im2, idim, ipar, con1, con2, con3, con4,    &
!!                    iorder1, iorder2, iopen1, iopen2, rsurf, jstat)
!!     Real(REAL64),    Intent(IN)    :: points(*)
!!     Integer,         Intent(IN)    :: im1
!!     Integer,         Intent(IN)    :: im2
!!     Integer,         Intent(IN)    :: idim
!!     Integer,         Intent(IN)    :: ipar
```

```
!!     Integer,         Intent(IN)    :: con1
!!     Integer,         Intent(IN)    :: con2
!!     Integer,         Intent(IN)    :: con3
!!     Integer,         Intent(IN)    :: con4
!!     Integer,         Intent(IN)    :: iorder1
!!     Integer,         Intent(IN)    :: iorder2
!!     Integer,         Intent(IN)    :: iopen1
!!     Integer,         Intent(IN)    :: iopen2
!!     Type(SISLsurf),  Intent(INOUT) :: rsurf
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s1537 -------------------------------------

  Subroutine s1537(points, im1, im2, idim, par1, par2, con1, con2, con3, con4, &
                   iorder1, iorder2, iopen1, iopen2, rsurf, jstat)

!! PURPOSE
!!   s1537 - Compute a tensor surface interpolating a set of points, parameter-
!!           ization as input. The output is represented as a B-spline surface.

!! INTERFACE
!!   Subroutine s1537(points, im1, im2, idim, par1, par2, con1, con2, con3,  &
!!                    con4, iorder1, iorder2, iopen1, iopen2, rsurf, jstat)
!!     Real(REAL64),    Intent(IN)    :: points(*)
!!     Integer,         Intent(IN)    :: im1
!!     Integer,         Intent(IN)    :: im2
!!     Integer,         Intent(IN)    :: idim
!!     Real(REAL64),    Intent(IN)    :: par1(*)
!!     Real(REAL64),    Intent(IN)    :: par2(*)
!!     Integer,         Intent(IN)    :: con1
!!     Integer,         Intent(IN)    :: con2
!!     Integer,         Intent(IN)    :: con3
!!     Integer,         Intent(IN)    :: con4
!!     Integer,         Intent(IN)    :: iorder1
!!     Integer,         Intent(IN)    :: iorder2
!!     Integer,         Intent(IN)    :: iopen1
!!     Integer,         Intent(IN)    :: iopen2
!!     Type(SISLsurf),  Intent(INOUT) :: rsurf
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s1534 -------------------------------------

  Subroutine s1534(points, der10, der01, der11, im1, im2, idim, ipar, con1,    &
                   con2, con3, con4, iorder1, iorder2, iopen1, iopen2, rsurf,  &
                   jstat)

!! PURPOSE
!!    s1534 - To compute a surface interpolating a set of points, derivatives as
!!            input. The output is represented as a B-spline surface.

!! INTERFACE
!!   Subroutine s1534(points, der10, der01, der11, im1, im2, idim, ipar, con1, &
!!                    con2, con3, con4, iorder1, iorder2, iopen1, iopen2,      &
!!                    rsurf, jstat)
!!     Real(REAL64),    Intent(IN)    :: points(*)
!!     Real(REAL64),    Intent(IN)    :: der10(*)
!!     Real(REAL64),    Intent(IN)    :: der01(*)
!!     Real(REAL64),    Intent(IN)    :: der11(*)
!!     Integer,         Intent(IN)    :: im1
```

```
!!     Integer,         Intent(IN)    :: im2
!!     Integer,         Intent(IN)    :: idim
!!     Integer,         Intent(IN)    :: ipar
!!     Integer,         Intent(IN)    :: con1
!!     Integer,         Intent(IN)    :: con2
!!     Integer,         Intent(IN)    :: con3
!!     Integer,         Intent(IN)    :: con4
!!     Integer,         Intent(IN)    :: iorder1
!!     Integer,         Intent(IN)    :: iorder2
!!     Integer,         Intent(IN)    :: iopen1
!!     Integer,         Intent(IN)    :: iopen2
!!     Type(SISLsurf),  Intent(INOUT) :: rsurf
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s1535 -------------------------------------

  Subroutine s1535(points, der10, der01, der11, im1, im2, idim, par1, par2,    &
                   con1, con2, con3, con4, iorder1, iorder2, rsurf, jstat)

!! PURPOSE
!!   s1535 - Compute a surface interpolating a set of points, derivatives and
!!           parameterization as input. The output is represented as a B-spline
!!           surface.

!! INTERFACE
!!   Subroutine s1535(points, der10, der01, der11, im1, im2, idim, par1, par2, &
!!                    con1, con2, con3, con4, iorder1, iorder2, rsurf, jstat)
!!     Real(REAL64),    Intent(IN)    :: points(*)
!!     Real(REAL64),    Intent(IN)    :: der10(*)
!!     Real(REAL64),    Intent(IN)    :: der01(*)
!!     Real(REAL64),    Intent(IN)    :: der11(*)
!!     Integer,         Intent(IN)    :: im1
!!     Integer,         Intent(IN)    :: im2
!!     Integer,         Intent(IN)    :: idim
!!     Real(REAL64),    Intent(IN)    :: par1(*)
!!     Real(REAL64),    Intent(IN)    :: par2(*)
!!     Integer,         Intent(IN)    :: con1
!!     Integer,         Intent(IN)    :: con2
!!     Integer,         Intent(IN)    :: con3
!!     Integer,         Intent(IN)    :: con4
!!     Integer,         Intent(IN)    :: iorder1
!!     Integer,         Intent(IN)    :: iorder2
!!     Type(SISLsurf),  Intent(INOUT) :: rsurf
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s1529 -------------------------------------

  Subroutine s1529(ep, eder10, eder01, eder11, im1, im2, idim, ipar, rsurf,    &
                   jstat)

!! PURPOSE
!!   s1529 - Compute the cubic Hermite surface interpolant to the data given.
!!           More specifically, given positions, (u',v), (u,v'), and (u',v')
!!           derivatives at points of a rectangular grid, the routine computes
!!           a cubic tensor-product B-spline interpolant to the given data with
!!           double knots at each data (the first knot vector will have double
!!           knots at all interior points in epar1, quadruple knots at the first
!!           and last points, and similarly for the second knot vector). The
!! output is represented as a B-spline surface.
```

```
!! INTERFACE
!!   Subroutine s1529(ep, eder10, eder01, eder11, im1, im2, idim, ipar, rsurf, &
!!                    jstat)
!!     Real(REAL64),    Intent(IN)    :: ep(*)
!!     Real(REAL64),    Intent(IN)    :: eder10(*)
!!     Real(REAL64),    Intent(IN)    :: eder01(*)
!!     Real(REAL64),    Intent(IN)    :: eder11(*)
!!     Integer,         Intent(IN)    :: im1
!!     Integer,         Intent(IN)    :: im2
!!     Integer,         Intent(IN)    :: idim
!!     Integer,         Intent(IN)    :: ipar
!!     Type(SISLsurf),  Intent(INOUT) :: rsurf
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s1530 -------------------------------------

  Subroutine s1530(ep, eder10, eder01, eder11, epar1, epar2, im1, im2, idim,   &
                   rsurf, jstat)

!! PURPOSE
!!   s1530 - To compute the cubic Hermite interpolant to the data given. More
!!           specifically, given positions, 10, 01, and 11 derivatives at
!!           points of a rectangular grid, the routine computes a cubic tensor-
!!           product B-spline interpolant to the given data with double knots
!!           at each data point (the first knot vector will have double knots at
!!           all interior points in epar1, quadruple knots at the first and last
!!           points, and similarly for the second knot vector). The output is
!!           represented as a B-spline surface.

!! INTERFACE
!!   Subroutine s1530(ep, eder10, eder01, eder11, epar1, epar2, im1, im2, idim,&
!!                    rsurf, jstat)
!!     Real(REAL64),    Intent(IN)    :: ep(*)
!!     Real(REAL64),    Intent(IN)    :: eder10(*)
!!     Real(REAL64),    Intent(IN)    :: eder01(*)
!!     Real(REAL64),    Intent(IN)    :: eder11(*)
!!     Real(REAL64),    Intent(IN)    :: epar1(*)
!!     Real(REAL64),    Intent(IN)    :: epar2(*)
!!     Integer,         Intent(IN)    :: im1
!!     Integer,         Intent(IN)    :: im2
!!     Integer,         Intent(IN)    :: idim
!!     Type(SISLsurf),  Intent(INOUT) :: rsurf
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s1538 -------------------------------------

  Subroutine s1538(inbcrv, vpcurv, nctyp, astpar, iopen, iord2, iflag, rsurf,  &
                   gpar, jstat)

!! PURPOSE
!!   s1538 - To create a lofted surface from a set of B-spline (i.e. NOT
!!           rational) input curves. The output is represented as a B-spline
!!           surface.

!! INTERFACE
!!   Subroutine s1538(inbcrv, vpcurv, nctyp, astpar, iopen, iord2, iflag,      &
!!                    rsurf, gpar, jstat)
!!     Integer,                        Intent(IN)    :: inbcrv
```

```
!!     Type(SISLcurve),                Intent(IN)    :: vpcurv(*)
!!     Integer,                        Intent(IN)    :: nctyp(*)
!!     Real(REAL64),                   Intent(IN)    :: astpar
!!     Integer,                        Intent(IN)    :: iopen
!!     Integer,                        Intent(IN)    :: iord2
!!     Integer,                        Intent(IN)    :: iflag
!!     Type(SISLsurf),                 Intent(INOUT) :: rsurf
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: gpar(:)
!!     Integer,                        Intent(INOUT) :: jstat


!---------------------------------- s1539 -------------------------------------

  Subroutine s1539(inbcrv, vpcurv, nctyp, epar, astpar, iopen, iord2, iflag,   &
                   rsurf, gpar, jstat)

!! PURPOSE
!!   s1539 - To create a spline lofted surface from a set of input curves. The
!!           parametrization of the position curves is given in epar.

!! INTERFACE
!!   Subroutine s1539(inbcrv, vpcurv, nctyp, epar, astpar, iopen, iord2,       &
!!                    iflag, rsurf, gpar, jstat)
!!     Integer,                        Intent(IN)    :: inbcrv
!!     Type(SISLcurve),                Intent(IN)    :: vpcurv(*)
!!     Integer,                        Intent(IN)    :: nctyp(*)
!!     Real(REAL64),                   Intent(IN)    :: epar(*)
!!     Real(REAL64),                   Intent(IN)    :: astpar
!!     Integer,                        Intent(IN)    :: iopen
!!     Integer,                        Intent(IN)    :: iord2
!!     Integer,                        Intent(IN)    :: iflag
!!     Type(SISLsurf),                 Intent(INOUT) :: rsurf
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: gpar(:)
!!     Integer,                        Intent(INOUT) :: jstat


!---------------------------------- s1508 -------------------------------------

  Subroutine s1508(inbcrv, vpcurv, par_arr, rsurf, jstat)

!! PURPOSE
!!   s1508 - To create a rational lofted surface from a set of rational input-
!!           curves

!! INTERFACE
!!   Subroutine s1508(inbcrv, vpcurv, par_arr, rsurf, jstat)
!!     Integer,                        Intent(IN)    :: inbcrv
!!     Type(SISLcurve),                Intent(IN)    :: vpcurv(*)
!!     Real(REAL64),                   Intent(IN)    :: par_arr(*)
!!     Type(SISLsurf),                 Intent(INOUT) :: rsurf
!!     Integer,                        Intent(INOUT) :: jstat

!---------------------------------- s1390 -------------------------------------

  Subroutine s1390(curves, rsurf, numder, stat)

!! PURPOSE
!! s1390 - Make a 4-edged blending surface between 4 B-spline (i.e. NOT
!!         rational) curves where each curve is associated with a number of
```

```
!!         cross-derivative B-spline (i.e. NOT rational) curves. The output is
!!         represented as a B-spline surface. The input curves are numbered
!1         successively around the blending parameter, and the directions
!!         of the curves are expected to be directed in the positive u or v
!!         directions with cross-derivatives always pointing into the patch

!! INTERFACE
!!   Subroutine s1390(curves, rsurf, numder, stat)
!!     Type(SISLcurve),                Intent(IN)    :: curves(:)
!!     Type(SISLsurf),                 Intent(INOUT) :: rsurf
!!     Integer,                        Intent(IN)    :: numder(4)
!!     Integer,                        Intent(INOUT) :: stat

!---------------------------------- s1391 -------------------------------------

  Subroutine s1391(pc, ws, icurv, nder, jstat)

!! PURPOSE
!!   s1391 - To create a first derivative continuous blending surface set
!!           over a 3-, 4-, 5- and 6-sided region in space. The boundary of the
!!           region are B-spline (i.e. NOT rational) curves and the cross
!!           boundary derivatives are given as B-spline (i.e. NOT rational)
!!           curves. This function automatically preprocesses the input cross
!!           tangent curves in order to make them suitable for the blending.
!!           Thus, the cross tangent curves should be taken as the cross tangents
!!           of the surrounding surface. It is not necessary and not advisable
!!           to match tangents etc. in the corners. The output is represented as
!!           a set of B-spline surfaces

!! INTERFACE
!!   Subroutine s1391(pc, ws, icurv, nder, jstat)
!!     Type(SISLcurve),              Intent(IN)    :: pc(:)
!!     Type(SISLsurf),  ALLOCATABLE, Intent(INOUT) :: ws(:)
!!     Integer,                      Intent(IN)    :: icurv
!!     Integer,                      Intent(IN)    :: nder(icurv)
!!     Integer,                      Intent(INOUT) :: jstat

!---------------------------------- s1401 -------------------------------------

  Subroutine s1401(vcurv, etwist, rsurf, jstat)

!! PURPOSE
!!   s1401 - Compute a Gordon patch, given position and cross tangent con-
!!           ditions as B-spline (i.e. NOT rational) curves at the boundary of
!!           a squared region and the twist vector in the corners. The output
!!           is represented as a B-spline surface.

!! INTERFACE
!!   Subroutine s1401(vcurv, etwist, rsurf, jstat)
!!     Type(SISLcurve), Intent(IN)    :: vcurv(8)
!!     Real(REAL64),    Intent(IN)    :: etwist(*)
!!     Type(SISLsurf),  Intent(INOUT) :: rsurf
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s1620 -------------------------------------

  Subroutine s1620(epoint, inbpnt1, inbpnt2, ipar, iopen1, iopen2, ik1, ik2,   &
                   idim, rs, jstat)
```

```
!! PURPOSE
!!   s1620 - To calculate a surface using the input points as control vertices.
!!           The parametrization is calculated according to ipar. The output
!!           is represented as a B-spline surface.

!! INTERFACE
!!   Subroutine s1620(epoint, inbpnt1, inbpnt2, ipar, iopen1, iopen2, ik1, ,   &
!!                    ik2, idim, rs, jstat)
!!     Real(REAL64),    Intent(IN)    :: epoint(*)
!!     Integer,         Intent(IN)    :: inbpnt1
!!     Integer,         Intent(IN)    :: inbpnt2
!!     Integer,         Intent(IN)    :: ipar
!!     Integer,         Intent(IN)    :: iopen1
!!     Integer,         Intent(IN)    :: iopen2
!!     Integer,         Intent(IN)    :: ik1
!!     Integer,         Intent(IN)    :: ik2
!!     Integer,         Intent(IN)    :: idim
!!     Type(SISLsurf),  Intent(INOUT) :: rs
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s1332 -------------------------------------

  Subroutine s1332(curve1, curve2, epsge, point, surf, stat)

!! PURPOSE
!!   s1332 - To create a linear swept surface by making the tensor-product of
!!           two curves.

!! INTERFACE
!!   Subroutine s1332(curve1, curve2, epsge, point, surf, stat)
!!     Type(SISLcurve), Intent(IN)    :: curve1
!!     Type(SISLcurve), Intent(IN)    :: curve2
!!     Real(REAL64),    Intent(IN)    :: epsge
!!     Real(REAL64),    Intent(IN)    :: point(*)
!!     Type(SISLsurf),  Intent(INOUT) :: surf
!!     Integer,         Intent(INOUT) :: stat

!---------------------------------- s1302 -------------------------------------

  Subroutine s1302(curve, epsge, angle, point, axis, surf, stat)

!! PURPOSE
!!   s1302 - To create a rotational swept surface by rotating a curve a given
!!           angle around the axis defined by point[ ] and axis[ ]. The maxi-
!!           mal deviation allowed between the true rotational surface and the
!!           generated surface, is epsge. If epsge is set to 0, a NURBS surface
!!           is generated and if epsge > 0, a B-spline surface is generated.

!! INTERFACE
!!   Subroutine s1302(curve, epsge, angle, point, axis, surf, stat)
!!     Type(SISLcurve), Intent(IN)    :: curve
!!     Real(REAL64),    Intent(IN)    :: epsge
!!     Real(REAL64),    Intent(IN)    :: angle
!!     Real(REAL64),    Intent(IN)    :: point(*)
!!     Real(REAL64),    Intent(IN)    :: axis(*)
!!     Type(SISLsurf),  Intent(INOUT) :: surf
!!     Integer,         Intent(INOUT) :: stat

!---------------------------------- s1365 -------------------------------------
```

```
  Subroutine s1365(ps, aoffset, aepsge, amax, idim, rs, jstat)

!! PURPOSE
!!   s1365 - Create a surface approximating the offset of a surface. The output
!!           is represented as a B-spline surface.
!!           With an offset of zero, this routine can be used to approximate any
!!           NURBS (rational) surface with a B-spline (non-rational) surface.

!! INTERFACE
!!   Subroutine s1365(ps, aepsge, amax, idim, rs, jstat)
!!     Type(SISLsurf), Intent(IN)    :: ps
!!     Real(REAL64),   Intent(IN)    :: aoffset
!!     Real(REAL64),   Intent(IN)    :: aepsge
!!     Real(REAL64),   Intent(IN)    :: amax
!!     Integer,        Intent(IN)    :: idim
!!     Type(SISLsurf), Intent(INOUT) :: rs
!!     Integer,        Intent(INOUT) :: jstat

!---------------------------------- s1601 -------------------------------------

  Subroutine s1601(psurf, epoint, enorm, idim, rsurf, stat)

!! PURPOSE
!!   s1601 - Mirror a surface about a plane.

!! INTERFACE
!!   Subroutine s1601(psurf, epoint, enorm, idim, rsurf, stat)
!!     Type(SISLsurf), Intent(IN)    :: psurf
!!     Real(REAL64),   Intent(IN)    :: epoint(*)
!!     Real(REAL64),   Intent(IN)    :: enorm(*)
!!     Integer,        Intent(IN)    :: idim
!!     Type(SISLsurf), Intent(INOUT) :: rsurf
!!     Integer,        Intent(INOUT) :: stat

!---------------------------------- s1388 -------------------------------------

  Subroutine s1388(surf, coons, numcoons1, numcoons2, dim, stat)

!! PURPOSE
!!   s1388 - To convert a surface of order less than or equal to 4 in both direc
!!           -tions to a mesh of Coons patches with uniform parameterization.
!!           This subroutine assumes that the surface is C1 continuous.

!! INTERFACE
!!   Subroutine s1388(surf, coons, numcoons1, numcoons2, dim, stat)
!!     Type(SISLsurf),              Intent(IN)    :: surf
!!     Real(REAL64),   ALLOCATABLE, Intent(INOUT) :: coons(:)
!!     Integer,                     Intent(INOUT) :: numcoons1
!!     Integer,                     Intent(INOUT) :: numcoons2
!!     Integer,                     Intent(INOUT) :: dim
!!     Integer,                     Intent(INOUT) :: stat

!---------------------------------- s1731 -------------------------------------

  Subroutine s1731(surf, newsurf, stat)

!! PURPOSE
!!   s1731 - To convert a surface to a mesh of Bezier surfaces. The Bezier
```

```
!!           surfaces are stored in a surface with all knots having multiplicity
!!           equal to the order of the surface in the corresponding parameter
!!           direction. If the input surface is rational, the generated Bezier
!!           surfaces will be rational too (i.e. there will be rational weights
!!           in the representation of the Bezier surfaces).

!! INTERFACE
!!   Subroutine s1731(surf, newsurf, stat)
!!     Type(SISLsurf), Intent(IN)    :: surf
!!     Type(SISLsurf), Intent(INOUT) :: newsurf
!!     Integer,        Intent(INOUT) :: stat

!---------------------------------- s1733 -----------------------------------

  Subroutine s1733(surf, number1, number2, startpar1, endpar1, startpar2,      &
                   endpar2, coef, stat)

!! PURPOSE
!!   s1733 - To pick the next Bezier surface from a surface. This function
!!           requires a surface represented as the result of s1731.
!!           This routine does not check that the surface is correct. If
!!           the input surface is rational, the generated Bezier surfaces will
!!           be rational too (i.e. there will be rational weights in the
!!           representation of the Bezier surfaces).

!! INTERFACE
!!   Subroutine s1733(surf, number1, number2, startpar1, endpar1, startpar2,   &
!!                    endpar2, coef, stat)
!!     Type(SISLsurf), Intent(IN)    :: surf
!!     Integer,        Intent(IN)    :: number1
!!     Integer,        Intent(IN)    :: number2
!!     Real(REAL64),   Intent(INOUT) :: startpar1
!!     Real(REAL64),   Intent(INOUT) :: endpar1
!!     Real(REAL64),   Intent(INOUT) :: startpar2
!!     Real(REAL64),   Intent(INOUT) :: endpar2
!!     Real(REAL64),   Intent(INOUT) :: coef(*)
!!     Integer,        Intent(INOUT) :: stat

!---------------------------------- s1387 -----------------------------------

  Subroutine s1387(surf, order1, order2, newsurf, stat)

!! PURPOSE
!!   s1387 - To express a surface as a surface of higher order.

!! INTERFACE
!!   Subroutine s1387(surf, order1, order2, newsurf, stat)
!!     Type(SISLsurf), Intent(IN)    :: surf
!!     Integer,        Intent(IN)    :: order1
!!     Integer,        Intent(IN)    :: order2
!!     Type(SISLsurf), Intent(INOUT) :: newsurf
!!     Integer,        Intent(INOUT) :: stat

!---------------------------------- s1386 -----------------------------------

  Subroutine s1386(surf, der1, der2, newsurf, stat)

!! PURPOSE
!!   s1386 - To express the (der1, der2)-th derivative of an open surface as a
```

```
!!          surface.

!! INTERFACE
!!   Subroutine s1386(surf, der1, der2, newsurf, stat)
!!     Type(SISLsurf), Intent(IN)    :: surf
!!     Integer,        Intent(IN)    :: der1
!!     Integer,        Intent(IN)    :: der2
!!     Type(SISLsurf), Intent(INOUT) :: newsurf
!!     Integer,        Intent(INOUT) :: stat

!---------------------------------- s1023 -------------------------------------

  Subroutine s1023(centre, axis, equator, latitude, longitude, sphere, stat)

!! PURPOSE
!!   s1023 - To express the octants of a sphere as a surface. This can also be
!!           used to describe the complete sphere. The sphere/the octants of
!!           the sphere will be geometrically exact.

!! INTERFACE
!!   Subroutine s1023(centre, axis, equator, latitude, longitude, sphere, stat)
!!     Real(REAL64),   Intent(IN)    :: centre(*)
!!     Real(REAL64),   Intent(IN)    :: axis(*)
!!     Real(REAL64),   Intent(IN)    :: equator(*)
!!     Integer,        Intent(IN)    :: latitude
!!     Integer,        Intent(IN)    :: longitude
!!     Type(SISLsurf), Intent(INOUT) :: sphere
!!     Integer,        Intent(INOUT) :: stat

!---------------------------------- s1021 -------------------------------------

  Subroutine s1021(bottom_pos, bottom_axis, ellipse_ratio, axis_dir, height,   &
                   cyl, stat)

!! PURPOSE
!!   s1021 - To express a truncated cylinder as a surface. The cylinder can be
!!           elliptic. The cylinder will be geometrically exact.

!! INTERFACE
!!   Subroutine s1021(bottom_pos, bottom_axis, ellipse_ratio, axis_dir, height,&
!!                    cyl, stat)
!!     Real(REAL64),   Intent(IN)    :: bottom_pos(*)
!!     Real(REAL64),   Intent(IN)    :: bottom_axis(*)
!!     Real(REAL64),   Intent(IN)    :: ellipse_ratio
!!     Real(REAL64),   Intent(IN)    :: axis_dir(*)
!!     Real(REAL64),   Intent(IN)    :: height
!!     Type(SISLsurf), Intent(INOUT) :: cyl
!!     Integer,        Intent(INOUT) :: stat

!---------------------------------- s1024 -------------------------------------

  Subroutine s1024(centre, axis, equator, minor_radius, start_minor,           &
                   end_minor, numb_major, torus, stat)

!! PURPOSE
!!   s1024 - To express the octants of a torus as a surface. This can also be
!!           used to describe the complete torus. The torus/the octants of the
!!           torus will be geometrically exact.
```

```
!! INTERFACE
!!   Subroutine s1024(centre, axis, equator, minor_radius, start_minor,       &
!!                    end_minor, numb_major, torus, stat)
!!     Real(REAL64),   Intent(IN)    :: centre(*)
!!     Real(REAL64),   Intent(IN)    :: axis(*)
!!     Real(REAL64),   Intent(IN)    :: equator(*)
!!     Real(REAL64),   Intent(IN)    :: minor_radius
!!     Integer,        Intent(IN)    :: start_minor
!!     Integer,        Intent(IN)    :: end_minor
!!     Integer,        Intent(IN)    :: numb_major
!!     Type(SISLsurf), Intent(INOUT) :: torus
!!     Integer,        Intent(INOUT) :: stat

!---------------------------------- s1022 -----------------------------------

  Subroutine s1022(bottom_pos, bottom_axis, ellipse_ratio, axis_dir,          &
                   cone_angle, height, cone, stat)

!! PURPOSE
!!   s1022 - To express a truncated cone as a surface. The cone can be elliptic.
!!           The cone will be geometrically exact.

!! INTERFACE
!!   Subroutine s1022(bottom_pos, bottom_axis, ellipse_ratio, axis_dir,       &
!!                    cone_angle, height, cone, stat)
!!     Real(REAL64),   Intent(IN)    :: bottom_pos(*)
!!     Real(REAL64),   Intent(IN)    :: bottom_axis(*)
!!     Real(REAL64),   Intent(IN)    :: ellipse_ratio
!!     Real(REAL64),   Intent(IN)    :: axis_dir(*)
!!     Real(REAL64),   Intent(IN)    :: cone_angle
!!     Real(REAL64),   Intent(IN)    :: height
!!     Type(SISLsurf), Intent(INOUT) :: cone
!!     Integer,        Intent(INOUT) :: stat
```

B.7 Surface Interrogation

```
!------------------------------- newIntCurve --------------------------------

  Subroutine newIntCurve(numgdpt, numpar1, numpar2, guidepar1, guidepar2,     &
                         itype, IntCurve)

!! PURPOSE
!!   newIntcurve - Create and initialize a SISLIntcurve-instance. Note that thes
!!   arrays guidepar1 and guidepar2 will be freed by freeIntcurve. In most
!!   cases the SISLIntcurve objects will be generated internally in the
!!   SISL intersection routines.

!! INTERFACE
!!   Subroutine newIntCurve(numgdpt, numpar1, numpar2, guidepar1, guidepar2,   &
!!                          itype, IntCurve)
!!     Integer,                  Intent(IN)    :: numgdpt
!!     Integer,                  Intent(IN)    :: numpar1
!!     Integer,                  Intent(IN)    :: numpar2
!!     Real(REAL64),             Intent(IN)    :: guidepar1(*)
!!     Real(REAL64),             Intent(IN)    :: guidepar2(*)
!!     Integer,                  Intent(IN)    :: itype
!!     Type(SISLIntcurve), TARGET, Intent(INOUT) :: IntCurve
```

```
!------------------------------ freeIntCurve --------------------------------

  Subroutine freeIntCurve(intCurve)

!! PURPOSE
!!   freeIntcurve - Free the space occupied by a SISLIntcurve.
!!   Note that the arrays guidepar1 and guidepar2 will be freed as well.

!! INTERFACE
!!   Subroutine freeIntCurve(intCurve)
!!     Type(SISLIntCurve), Target, Intent(INOUT) :: intCurve


!------------------------------ freeIntCrvlist ------------------------------

  Subroutine freeIntCrvlist(IntCurve, icrv)

!! PURPOSE
!!   freeIntcrvlist - Free a list of SISLIntcurve.

!! INTERFACE
!!   Subroutine freeIntCrvlist(IntCurve, icrv)
!!     Type(SISLIntCurve), Target, Intent(INOUT) :: IntCurve(*)
!!     Integer,                    Intent(IN)    :: icrv

! S1850 is same routine as in Curve_Interrogation module
! S1371 is same routine as in Curve_Interrogation module

!---------------------------------- s1372 -----------------------------------

  Subroutine s1372(curve, point, dir, radius, dim, epsco, epsge, numintpt,     &
                   intpar, numintcu, intcurve, stat)

!! PURPOSE
!!   s1372 - Find all the intersections between a curve and a cylinder.

!! INTERFACE
!!   Subroutine s1372(curve, point, dir, radius, dim, epsco, epsge, numintpt, &
!!                    intpar, numintcu, intcurve, stat)
!!     Type(SISLcurve),                Intent(IN)    :: curve
!!     Real(REAL64),                   Intent(IN)    :: point(*)
!!     Real(REAL64),                   Intent(IN)    :: dir(*)
!!     Real(REAL64),                   Intent(IN)    :: radius
!!     Integer,                        Intent(IN)    :: dim
!!     Real(REAL64),                   Intent(IN)    :: epsco
!!     Real(REAL64),                   Intent(IN)    :: epsge
!!     Integer,                        Intent(INOUT) :: numintpt
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: intpar(:)
!!     Integer,                        Intent(INOUT) :: numintcu
!!     Type(SISLIntcurve), ALLOCATABLE, Intent(INOUT) :: intcurve(:)
!!     Integer,                        Intent(INOUT) :: stat


!---------------------------------- s1373 -----------------------------------

  Subroutine s1373(curve, top, axispt, conept, dim, epsco, epsge, numintpt,    &
                   intpar, numintcu, intcurve, stat)
```

```
!! PURPOSE
!!   s1373 - Find all the intersections between a curve and a cone.

!! INTERFACE
!!   Subroutine s1373(curve, top, axispt, conept, dim, epsco, epsge, numintpt, &
!!                    intpar, numintcu, intcurve, stat)
!!     Type(SISLcurve),                  Intent(IN)    :: curve
!!     Real(REAL64),                     Intent(IN)    :: top(*)
!!     Real(REAL64),                     Intent(IN)    :: axispt(*)
!!     Real(REAL64),                     Intent(IN)    :: conept(*)
!!     Integer,                          Intent(IN)    :: dim
!!     Real(REAL64),                     Intent(IN)    :: epsco
!!     Real(REAL64),                     Intent(IN)    :: epsge
!!     Integer,                          Intent(INOUT) :: numintpt
!!     Real(REAL64),       ALLOCATABLE,  Intent(INOUT) :: intpar(:)
!!     Integer,                          Intent(INOUT) :: numintcu
!!     Type(SISLIntcurve), ALLOCATABLE,  Intent(INOUT) :: intcurve(:)
!!     Integer,                          Intent(INOUT) :: stat


!---------------------------------- s1502 -------------------------------------

  Subroutine s1502(curve, basept, normdir, ellipaxis, alpha, ratio, dim,       &
                   epsco, epsge, numintpt, intpar, numintcu, intcurve, stat)

!! PURPOSE
!!   s1502 - Find all the intersections between a curve and an elliptic cone.

!! INTERFACE
!!   Subroutine s1502(curve, basept, normdir, ellipaxis, alpha, ratio, dim,  &
!!                    epsco, epsge, numintpt, intpar, numintcu, intcurve,    &
!!                    stat)
!!     Type(SISLcurve),                  Intent(IN)    :: curve
!!     Real(REAL64),                     Intent(IN)    :: basept(*)
!!     Real(REAL64),                     Intent(IN)    :: normdir(*)
!!     Real(REAL64),                     Intent(IN)    :: ellipaxis(*)
!!     Real(REAL64),                     Intent(IN)    :: alpha
!!     Real(REAL64),                     Intent(IN)    :: ratio
!!     Integer,                          Intent(IN)    :: dim
!!     Real(REAL64),                     Intent(IN)    :: epsco
!!     Real(REAL64),                     Intent(IN)    :: epsge
!!     Integer,                          Intent(INOUT) :: numintpt
!!     Real(REAL64),       ALLOCATABLE,  Intent(INOUT) :: intpar(:)
!!     Integer,                          Intent(INOUT) :: numintcu
!!     Type(SISLIntcurve), ALLOCATABLE,  Intent(INOUT) :: intcurve(:)
!!     Integer,                          Intent(INOUT) :: stat


!---------------------------------- s1375 -------------------------------------

  Subroutine s1375(curve, centre, normal, centdis, rad, dim, epsco, epsge,     &
                   numintpt, intpar, numintcu, intcurve, stat)

!! PURPOSE
!!   s1375 - Find all the intersections between a curve and a torus

!! INTERFACE
!!   Subroutine s1375(curve, centre, normal, centdis, rad, dim, epsco, epsge, &
!!                    numintpt, intpar, numintcu, intcurve, stat)
```

```
!!     Type(SISLcurve),                 Intent(IN)    :: curve
!!     Real(REAL64),                    Intent(IN)    :: centre(*)
!!     Real(REAL64),                    Intent(IN)    :: normal(*)
!!     Real(REAL64),                    Intent(IN)    :: centdis
!!     Real(REAL64),                    Intent(IN)    :: rad
!!     Integer,                         Intent(IN)    :: dim
!!     Real(REAL64),                    Intent(IN)    :: epsco
!!     Real(REAL64),                    Intent(IN)    :: epsge
!!     Integer,                         Intent(INOUT) :: numintpt
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: intpar(:)
!!     Integer,                         Intent(INOUT) :: numintcu
!!     Type(SISLIntcurve), ALLOCATABLE, Intent(INOUT) :: intcurve(:)
!!     Integer,                         Intent(INOUT) :: stat

!---------------------------------- s1870 ------------------------------------

  Subroutine s1870(ps1, pt1, idim, aepsge, jpt, gpar1, jcrv, wcurve, jstat)

!! PURPOSE
!!   s1870 - Find all intersections between a surface and a point.

!! INTERFACE
!!   Subroutine s1870(ps1, pt1, idim, aepsge, jpt, gpar1, jcrv, wcurve, jstat)
!!     Type(SISLsurf),                  Intent(IN)    :: ps1
!!     Real(REAL64),                    Intent(IN)    :: pt1(*)
!!     Integer,                         Intent(IN)    :: idim
!!     Real(REAL64),                    Intent(IN)    :: aepsge
!!     Integer,                         Intent(INOUT) :: jpt
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: gpar1(:)
!!     Integer,                         Intent(INOUT) :: jcrv
!!     Type(SISLIntcurve), ALLOCATABLE, Intent(INOUT) :: wcurve(:)
!!     Integer,                         Intent(INOUT) :: jstat

!---------------------------------- s1856 ------------------------------------

  Subroutine s1856(surf, point, linedir, dim, epsco, epsge, numintpt,          &
                   pointpar, numintcr, intcurves, stat)

!! PURPOSE
!!   s1856 - Find all intersections between a tensor-product surface and an
!!           infinite straight line.

!! INTERFACE
!!   Subroutine s1856(surf, point, linedir, dim, epsco, epsge, numintpt,      &
!!                    pointpar, numintcr, intcurves, stat)
!!     Type(SISLsurf),                  Intent(IN)    :: surf
!!     Real(REAL64),                    Intent(IN)    :: point(*)
!!     Real(REAL64),                    Intent(IN)    :: linedir(*)
!!     Integer,                         Intent(IN)    :: dim
!!     Real(REAL64),                    Intent(IN)    :: epsco
!!     Real(REAL64),                    Intent(IN)    :: epsge
!!     Integer,                         Intent(INOUT) :: numintpt
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: pointpar(:)
!!     Integer,                         Intent(INOUT) :: numintcr
!!     Type(SISLIntcurve), ALLOCATABLE, Intent(INOUT) :: intcurves(:)
!!     Integer,                         Intent(INOUT) :: stat


!---------------------------------- s1518 ------------------------------------
```

```
  Subroutine s1518(surf, point, dir, epsge, start, end, parin, parout, stat)

!! PURPOSE
!!   s1518 - Newton iteration on the intersection between a 3D NURBS surface
!!           and a line. If a good initial guess is given, the intersection will
!!           be found quickly. However if a bad initial guess is given, the
!!           iteration might not converge. We only search in the rectangular
!!           subdomain specified by ”start” and ”end”. This can be the whole
!!           domain if desired.

!! INTERFACE
!!   Subroutine s1518(surf, point, dir, epsge, start, end, parin, parout, stat)
!!     Type(SISLsurf),  Intent(IN)    :: surf
!!     Real(REAL64),    Intent(IN)    :: point(*)
!!     Real(REAL64),    Intent(IN)    :: dir(*)
!!     Real(REAL64),    Intent(IN)    :: epsge
!!     Real(REAL64),    Intent(IN)    :: start(*)
!!     Real(REAL64),    Intent(IN)    :: end(*)
!!     Real(REAL64),    Intent(IN)    :: parin(*)
!!     Real(REAL64),    Intent(INOUT) :: parout(*)
!!     Integer,         Intent(INOUT) :: stat


!---------------------------------- 1328 -------------------------------------

  Subroutine s1328(psold, epoint, enorm1, enorm2, idim, rsnew, jstat)

!! PURPOSE
!!   s1328 - Put the equation of the surface pointed at by psold into two planes
!!           given by the point epoint and the normals enorm1 and enorm2.
!!           The result is an equation where the new two-dimensional surface
!!           rsnew is to be equal to origo.

!! INTERFACE
!!   Subroutine s1328(psold, epoint, enorm1, enorm2, idim, rsnew, jstat)
!!     Type(SISLsurf), Intent(IN)    :: psold
!!     Real(REAL64),   Intent(IN)    :: epoint(*)
!!     Real(REAL64),   Intent(IN)    :: enorm1(*)
!!     Real(REAL64),   Intent(IN)    :: enorm2(*)
!!     Integer,        Intent(IN)    :: idim
!!     Type(SISLsurf), Intent(INOUT) :: rsnew
!!     Integer,        Intent(INOUT) :: jstat


!---------------------------------- s1855 ------------------------------------

  Subroutine s1855(surf, centre, radius, normal, dim, epsco, epsge, numintpt,  &
                   pointpar, numintcr, intcurves, stat)

!! PURPOSE
!!   s1855 - Find all intersections between a tensor-product surface and a full
!!           circle.

!! INTERFACE
!!   Subroutine s1855(surf, centre, radius, normal, dim, epsco, epsge,         &
!!                    numintpt, pointpar, numintcr, intcurves, stat)
!!     Type(SISLsurf),                  Intent(IN)    :: surf
!!     Real(REAL64),                    Intent(IN)    :: centre(*)
```

```
!!     Real(REAL64),                    Intent(IN)    :: radius
!!     Real(REAL64),                    Intent(IN)    :: normal(*)
!!     Integer,                         Intent(IN)    :: dim
!!     Real(REAL64),                    Intent(IN)    :: epsco
!!     Real(REAL64),                    Intent(IN)    :: epsge
!!     Integer,                         Intent(INOUT) :: numintpt
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: pointpar(:)
!!     Integer,                         Intent(INOUT) :: numintcr
!!     Type(SISLIntcurve), ALLOCATABLE, Intent(INOUT) :: intcurves(:)
!!     Integer,                         Intent(INOUT) :: stat


!---------------------------------- s1858 -------------------------------------

  Subroutine s1858(surf, curve, epsco, epsge, numintpt, pointpar1, pointpar2,  &
                   numintcr, intcurves, stat)

!! PURPOSE
!!   s1858 - Find all intersections between a surface and a curve. Intersection
!!           curves are described by guide points. To pick the intersection
!!           curves use s1712

!! INTERFACE
!!   Subroutine s1858(surf, curve, epsco, epsge, numintpt, pointpar1,         &
!!                    pointpar2, numintcr, intcurves, stat)
!!     Type(SISLsurf),                  Intent(IN)    :: surf
!!     Type(SISLcurve),                 Intent(IN)    :: curve
!!     Real(REAL64),                    Intent(IN)    :: epsco
!!     Real(REAL64),                    Intent(IN)    :: epsge
!!     Integer,                         Intent(INOUT) :: numintpt
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: pointpar1(:)
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: pointpar2(:)
!!     Integer,                         Intent(INOUT) :: numintcr
!!     Type(SISLIntcurve), ALLOCATABLE, Intent(INOUT) :: intcurves(:)
!!     Integer,                         Intent(INOUT) :: stat


!---------------------------------- s1851 -------------------------------------

  Subroutine s1851(surf, point, normal, dim, epsco, epsge, numintpt,          &
                   pointpar, numintcr, intcurves, stat)

!! PURPOSE
!!   s1851 - Find all intersections between a tensor-product surface and a
!!           plane. Intersection curves are described by guide points. To make
!!           the intersection curves use s1314.

!! INTERFACE
!!   Subroutine s1851(surf, point, normal, dim, epsco, epsge, numintpt,       &
!!                    pointpar, numintcr, intcurves, stat)
!!     Type(SISLsurf),                  Intent(IN)    :: surf
!!     Real(REAL64),                    Intent(IN)    :: point(*)
!!     Real(REAL64),                    Intent(IN)    :: normal(*)
!!     Integer,                         Intent(IN)    :: dim
!!     Real(REAL64),                    Intent(IN)    :: epsco
!!     Real(REAL64),                    Intent(IN)    :: epsge
!!     Integer,                         Intent(INOUT) :: numintpt
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: pointpar(:)
!!     Integer,                         Intent(INOUT) :: numintcr
```

```
!!     Type(SISLIntcurve), ALLOCATABLE, Intent(INOUT) :: intcurves(:)
!!     Integer,                         Intent(INOUT) :: stat


!---------------------------------- s1852 -------------------------------------

  Subroutine s1852(surf, centre, radius, dim, epsco, epsge, numintpt,          &
                   pointpar, numintcr, intcurves, stat)

!! PURPOSE
!!   s1852 - Find all intersections between a tensor-product surface and a
!!           sphere. Intersection curves are described by guide points. To pro-
!!           duce the intersection curves use s1315.

!! INTERFACE
!!   Subroutine s1852(surf, centre, radius, dim, epsco, epsge, numintpt,      &
!!                    pointpar, numintcr, intcurves, stat)
!!     Type(SISLsurf),                  Intent(IN)    :: surf
!!     Real(REAL64),                    Intent(IN)    :: centre(*)
!!     Real(REAL64),                    Intent(IN)    :: radius
!!     Integer,                         Intent(IN)    :: dim
!!     Real(REAL64),                    Intent(IN)    :: epsco
!!     Real(REAL64),                    Intent(IN)    :: epsge
!!     Integer,                         Intent(INOUT) :: numintpt
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: pointpar(:)
!!     Integer,                         Intent(INOUT) :: numintcr
!!     Type(SISLIntcurve), ALLOCATABLE, Intent(INOUT) :: intcurves(:)
!!     Integer,                         Intent(INOUT) :: stat


!---------------------------------- s1853 -------------------------------------

  Subroutine s1853(surf, point, cyldir, radius, dim, epsco, epsge, numintpt,   &
                   pointpar, numintcr, intcurve, stat)

!! PURPOSE
!!   s1853 - Find all intersections between a tensor-product surface and a
!!           cylinder. Intersection curves are described by guide points. To
!!           produce the intersection curves use s1316.

!! INTERFACE
!!   Subroutine s1853(surf, point, cyldir, radius, dim, epsco, epsge, numintpt,&
!!                    pointpar, numintcr, intcurve, stat)
!!     Type(SISLsurf),                  Intent(IN)    :: surf
!!     Real(REAL64),                    Intent(IN)    :: point(*)
!!     Real(REAL64),                    Intent(IN)    :: cyldir(*)
!!     Real(REAL64),                    Intent(IN)    :: radius
!!     Integer,                         Intent(IN)    :: dim
!!     Real(REAL64),                    Intent(IN)    :: epsco
!!     Real(REAL64),                    Intent(IN)    :: epsge
!!     Integer,                         Intent(INOUT) :: numintpt
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: pointpar(:)
!!     Integer,                         Intent(INOUT) :: numintcr
!!     Type(SISLIntcurve), ALLOCATABLE, Intent(INOUT) :: intcurve(:)
!!     Integer,                         Intent(INOUT) :: stat


!---------------------------------- s1854 -------------------------------------
```

```
  Subroutine s1854(surf, toppt, axispt, conept, dim, epsco, epsge, numintpt,   &
                   pointpar, numintcr, intcurve, stat)

!! PURPOSE
!!   s1854 - Find all intersections between a tensor-product surface and a cone.
!!           Intersection curves are described by guide points. To produce the
!!           intersection curves use s1317.

!! INTERFACE
!!   Subroutine s1854(surf, toppt, axispt, conept, dim, epsco, epsge, numintpt,&
!!                    pointpar, numintcr, intcurve, stat)
!!     Type(SISLsurf),                  Intent(IN)    :: surf
!!     Real(REAL64),                    Intent(IN)    :: toppt(*)
!!     Real(REAL64),                    Intent(IN)    :: axispt(*)
!!     Real(REAL64),                    Intent(IN)    :: conept(*)
!!     Integer,                         Intent(IN)    :: dim
!!     Real(REAL64),                    Intent(IN)    :: epsco
!!     Real(REAL64),                    Intent(IN)    :: epsge
!!     Integer,                         Intent(INOUT) :: numintpt
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: pointpar(:)
!!     Integer,                         Intent(INOUT) :: numintcr
!!     Type(SISLIntcurve), ALLOCATABLE, Intent(INOUT) :: intcurve(:)
!!     Integer,                         Intent(INOUT) :: stat

!---------------------------------- s1503 -------------------------------------

  Subroutine s1503(surf, basept, normdir, ellipaxis, alpha, ratio, dim,        &
                   epsco, epsge, numintpt, pointpar, numintcr, intcurve, stat)

!! PURPOSE
!!   s1503 - Find all intersections between a tensor-product surface and an
!!           elliptic cone. Intersection curves are described by guide points.
!!           To produce the intersection curves use s1501.

!! INTERFACE
!!   Subroutine s1503(surf, basept, normdir, ellipaxis, alpha, ratio, dim,    &
!!                    epsco, epsge, numintpt, pointpar, numintcr, intcurve,   &
!!                    stat)
!!     Type(SISLsurf),                  Intent(IN)    :: surf
!!     Real(REAL64),                    Intent(IN)    :: basept(*)
!!     Real(REAL64),                    Intent(IN)    :: normdir(*)
!!     Real(REAL64),                    Intent(IN)    :: ellipaxis(*)
!!     Real(REAL64),                    Intent(IN)    :: alpha
!!     Real(REAL64),                    Intent(IN)    :: ratio
!!     Integer,                         Intent(IN)    :: dim
!!     Real(REAL64),                    Intent(IN)    :: epsco
!!     Real(REAL64),                    Intent(IN)    :: epsge
!!     Integer,                         Intent(INOUT) :: numintpt
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: pointpar(:)
!!     Integer,                         Intent(INOUT) :: numintcr
!!     Type(SISLIntcurve), ALLOCATABLE, Intent(INOUT) :: intcurve(:)

!---------------------------------- s1369 -------------------------------------

  Subroutine s1369(surf, centre, normal, cendis, rad, dim, epsco, epsge,       &
                   numintpt, pointpar, numintcr, intcurve, stat)

!! PURPOSE
!!   s1369 - Find all intersections between a surface and a torus. Intersection
```

```
!!           curves are described by guide points. To produce the intersection
!!           curves use s1318.

!! INTERFACE
!!   Subroutine s1369(surf, centre, normal, cendis, rad, dim, epsco, epsge,   &
!!                    numintpt, pointpar, numintcr, intcurve, stat)
!!     Type(SISLsurf),                  Intent(IN)    :: surf
!!     Real(REAL64),                    Intent(IN)    :: centre(*)
!!     Real(REAL64),                    Intent(IN)    :: normal(*)
!!     Real(REAL64),                    Intent(IN)    :: cendis
!!     Real(REAL64),                    Intent(IN)    :: rad
!!     Integer,                         Intent(IN)    :: dim
!!     Real(REAL64),                    Intent(IN)    :: epsco
!!     Real(REAL64),                    Intent(IN)    :: epsge
!!     Integer,                         Intent(INOUT) :: numintpt
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: pointpar(:)
!!     Integer,                         Intent(INOUT) :: numintcr
!!     Type(SISLIntcurve), ALLOCATABLE, Intent(INOUT) :: intcurve(:)
!!     Integer,                         Intent(INOUT) :: stat

!---------------------------------- s1859 -------------------------------------

  Subroutine s1859(surf1, surf2, epsco, epsge, numintpt, pointpar1, pointpar2, &
                   numintcr, intcurves, stat)

!! PURPOSE
!!   s1859 - Find all intersections between two surfaces. Intersection curves
!!           are described by guide points. To produce the intersection curves
!!           use s1310.

!! INTERFACE
!!   Subroutine s1859(surf1, surf2, epsco, epsge, numintpt, pointpar1,        &
!!                    pointpar2, numintcr, intcurves, stat)
!!     Type(SISLsurf),                  Intent(IN)    :: surf1
!!     Type(SISLsurf),                  Intent(IN)    :: surf2
!!     Real(REAL64),                    Intent(IN)    :: epsco
!!     Real(REAL64),                    Intent(IN)    :: epsge
!!     Integer,                         Intent(INOUT) :: numintpt
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: pointpar1(:)
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: pointpar2(:)
!!     Integer,                         Intent(INOUT) :: numintcr
!!     Type(SISLIntcurve), ALLOCATABLE, Intent(INOUT) :: intcurves(:)
!!     Integer,                         Intent(INOUT) :: stat


!---------------------------------- s1860 -------------------------------------

  Subroutine s1860(surf, viewdir, dim, epsco, epsge, numsilpt, pointpar,       &
                   numsilcr, silcurves, stat)

!! PURPOSE
!!   s1860 - Find the silhouette curves and points of a surface when the surface
!!           is viewed from a specific direction (i.e. parallel projection). In
!!           addition to the points and curves found by this routine, break
!!           curves and edge-curves might be silhouette curves. Silhouette
!!           curves are described by guide points. To produce the silhouette
!!           curves use s1319.

!! INTERFACE
```

```
!!   Subroutine s1860(surf, viewdir, dim, epsco, epsge, numsilpt, pointpar,   &
!!                    numsilcr, silcurves, stat)
!!     Type(SISLsurf),                  Intent(IN)    :: surf
!!     Real(REAL64),                    Intent(IN)    :: viewdir(*)
!!     Integer,                         Intent(IN)    :: dim
!!     Real(REAL64),                    Intent(IN)    :: epsco
!!     Real(REAL64),                    Intent(IN)    :: epsge
!!     Integer,                         Intent(INOUT) :: numsilpt
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: pointpar(:)
!!     Integer,                         Intent(INOUT) :: numsilcr
!!     Type(SISLIntcurve), ALLOCATABLE, Intent(INOUT) :: silcurves(:)
!!     Integer,                         Intent(INOUT) :: stat


!---------------------------------- s1510 -------------------------------------

  Subroutine s1510(ps, eyepoint, idim, aepsco, aepsge, jpt, gpar, jcrv,        &
                   wcurve, jstat)

!! PURPOSE

!!   s1510 - Find the silhouette curves and points of a surface when the surface
!!           is viewed perspectively from a specific eye point. In addition to
!!           the points and curves found by this routine, break curves and edge-
!!           curves might be silhouette curves. To march out the silhouette
!!           curves, use s1514.

!! INTERFACE
!!   Subroutine s1510(ps, eyepoint, idim, aepsco, aepsge, jpt, gpar, jcrv,   &
!!                    wcurve, jstat)
!!     Type(SISLsurf),                  Intent(IN)    :: ps
!!     Real(REAL64),                    Intent(IN)    :: eyepoint(*)
!!     Integer,                         Intent(IN)    :: idim
!!     Real(REAL64),                    Intent(IN)    :: aepsco
!!     Real(REAL64),                    Intent(IN)    :: aepsge
!!     Integer,                         Intent(INOUT) :: jpt
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: gpar(:)
!!     Integer,                         Intent(INOUT) :: jcrv
!!     Type(SISLIntcurve), ALLOCATABLE, Intent(INOUT) :: wcurve(:)
!!     Integer,                         Intent(INOUT) :: jstat


!---------------------------------- s1511 -------------------------------------

  Subroutine s1511(ps, qpoint, bvec, idim, aepsco, aepsge, jpt, gpar, jcrv,    &
                   wcurve, jstat)

!! PURPOSE
!!   s1511 - Find the circular silhouette curves and points of a surface. In
!!           addition to the points and curves found by this routine, break
!!           curves and edge-curves might be silhouette curves. To march out
!!           the silhouette curves use s1515.

!! INTERFACE
!!   Subroutine s1511(ps, qpoint, bvec, idim, aepsco, aepsge, jpt, gpar, jcrv, &
!!                    wcurve, jstat)
!!     Type(SISLsurf),                  Intent(IN)    :: ps
!!     Real(REAL64),                    Intent(IN)    :: qpoint(*)
!!     Real(REAL64),                    Intent(IN)    :: bvec(*)
```

```
!!     Integer,                          Intent(IN)    :: idim
!!     Real(REAL64),                     Intent(IN)    :: aepsco
!!     Real(REAL64),                     Intent(IN)    :: aepsge
!!     Integer,                          Intent(INOUT) :: jpt
!!     Real(REAL64),       ALLOCATABLE,  Intent(INOUT) :: gpar(:)
!!     Integer,                          Intent(INOUT) :: jcrv
!!     Type(SISLIntcurve), ALLOCATABLE,  Intent(INOUT) :: wcurve(:)
!!     Integer,                          Intent(INOUT) :: jstat


!---------------------------------- s1314 -------------------------------------

  Subroutine s1314(surf, point, normal, dim, epsco, epsge, maxstep, intcurve,  &
                   makecurv, graphic, stat)

!! PURPOSE
!!   s1314 - To march an intersection curve described by parameter pairs in an
!!           intersection curve object, a surface and a plane. The guide points
!!           are expected to be found by s1851. The generated geometric curves
!!           are represented as B-spline curves.

!! INTERFACE
!!   Subroutine s1314(surf, point, normal, dim, epsco, epsge, maxstep,        &
!!                    intcurve, makecurv, graphic, stat)
!!     Type(SISLsurf),                 Intent(IN)    :: surf
!!     Real(REAL64),                   Intent(IN)    :: point(*)
!!     Real(REAL64),                   Intent(IN)    :: normal(*)
!!     Integer,                        Intent(IN)    :: dim
!!     Real(REAL64),                   Intent(IN)    :: epsco
!!     Real(REAL64),                   Intent(IN)    :: epsge
!!     Real(REAL64),                   Intent(IN)    :: maxstep
!!     Type(SISLIntcurve),             Intent(INOUT) :: intcurve
!!     Integer,                        Intent(IN)    :: makecurv
!!     Integer,                        Intent(IN)    :: graphic
!!     Integer,                        Intent(INOUT) :: stat


!---------------------------------- s1315 -------------------------------------

  Subroutine s1315(surf, centre, radius, dim, epsco, epsge, maxstep, intcurve, &
                   makecurv, graphic, stat)

!! PURPOSE
!!   s1315 - To march an intersection curve described by parameter pairs in an
!!           intersection curve object, a surface and a sphere. The guide points
!!           are expected to be found by s1852. The generated geometric curves
!!           are represented as B-spline curves.

!! INTERFACE
!!   Subroutine s1315(surf, centre, radius, dim, epsco, epsge, maxstep,       &
!!                    intcurve, makecurv, graphic, stat)
!!     Type(SISLsurf),                 Intent(IN)    :: surf
!!     Real(REAL64),                   Intent(IN)    :: centre(*)
!!     Real(REAL64),                   Intent(IN)    :: radius
!!     Integer,                        Intent(IN)    :: dim
!!     Real(REAL64),                   Intent(IN)    :: epsco
!!     Real(REAL64),                   Intent(IN)    :: epsge
!!     Real(REAL64),                   Intent(IN)    :: maxstep
!!     Type(SISLIntcurve),             Intent(INOUT) :: intcurve
```

```
!!     Integer,                          Intent(IN)    :: makecurv
!!     Integer,                          Intent(IN)    :: graphic
!!     Integer,                          Intent(INOUT) :: stat


!---------------------------------- s1316 -------------------------------------

  Subroutine s1316(surf, point, cyldir, radius, dim, epsco, epsge, maxstep,    &
                   intcurve, makecurv, graphic, stat)

!! PURPOSE
!!   1316 - To march an intersection curve described by parameter pairs in
!!          an intersection curve object, a surface and a cylinder. The guide
!!          points are expected to be found by s1853. The generated geometric
!!          curves are represented as B-spline curves.

!! INTERFACE
!!   Subroutine s1316(surf, point, cyldir, radius, dim, epsco, epsge,        &
!!                    maxstep, intcurve, makecurv, graphic, stat)
!!     Type(SISLsurf),                   Intent(IN)    :: surf
!!     Real(REAL64),                     Intent(IN)    :: point(*)
!!     Real(REAL64),                     Intent(IN)    :: cyldir(*)
!!     Real(REAL64),                     Intent(IN)    :: radius
!!     Integer,                          Intent(IN)    :: dim
!!     Real(REAL64),                     Intent(IN)    :: epsco
!!     Real(REAL64),                     Intent(IN)    :: epsge
!!     Real(REAL64),                     Intent(IN)    :: maxstep
!!     Type(SISLIntcurve),               Intent(INOUT) :: intcurve
!!     Integer,                          Intent(IN)    :: makecurv
!!     Integer,                          Intent(IN)    :: graphic
!!     Integer,                          Intent(INOUT) :: stat


!---------------------------------- s1317 -------------------------------------

  Subroutine s1317(surf, toppt, axispt, conept, dim, epsco, epsge, maxstep,    &
                   intcurve, makecurv, graphic, stat)

!! PURPOSE
!!   s1317 - To march an intersection curve described by parameter pairs in an
!!           intersection curve object, a surface and a cone. The guide points
!!           are expected to be found by s1854. The generated geometric
!!           curves are represented as B-spline curves.

!! INTERFACE
!!   Subroutine s1317(surf, toppt, axispt, conept, dim, epsco, epsge,        &
!!                    maxstep, intcurve, makecurv, graphic, stat)
!!     Type(SISLsurf),                   Intent(IN)    :: surf
!!     Real(REAL64),                     Intent(IN)    :: toppt(*)
!!     Real(REAL64),                     Intent(IN)    :: axispt(*)
!!     Real(REAL64),                     Intent(IN)    :: conept(*)
!!     Integer,                          Intent(IN)    :: dim
!!     Real(REAL64),                     Intent(IN)    :: epsco
!!     Real(REAL64),                     Intent(IN)    :: epsge
!!     Real(REAL64),                     Intent(IN)    :: maxstep
!!     Type(SISLIntcurve),               Intent(INOUT) :: intcurve
!!     Integer,                          Intent(IN)    :: makecurv
!!     Integer,                          Intent(IN)    :: graphic
!!     Integer,                          Intent(INOUT) :: stat
```

```
!---------------------------------- s1501 -------------------------------------

  Subroutine s1501(surf, basept, normdir, ellipaxis, alpha, ratio, dim, epsco, &
                   epsge, maxstep, intcurve, makecurv, graphic, stat)

!! PURPOSE
!!   s1501 - To march an intersection curve described by parameter pairs in
!!           an intersection curve object, a surface and an elliptic cone. The
!!           guide points are expected to be found by s1503. The generated
!!           geometric curves are represented as B-spline curves.

!! INTERFACE
!!   Subroutine s1501(surf, basept, normdir, ellipaxis, alpha, ratio, dim,  &
!!                    epsco, epsge, maxstep, intcurve, makecurv, graphic,   &
!!                    stat)
!!     Type(SISLsurf),                 Intent(IN)    :: surf
!!     Real(REAL64),                   Intent(IN)    :: basept(*)
!!     Real(REAL64),                   Intent(IN)    :: normdir(*)
!!     Real(REAL64),                   Intent(IN)    :: ellipaxis(*)
!!     Real(REAL64),                   Intent(IN)    :: alpha
!!     Real(REAL64),                   Intent(IN)    :: ratio
!!     Integer,                        Intent(IN)    :: dim
!!     Real(REAL64),                   Intent(IN)    :: epsco
!!     Real(REAL64),                   Intent(IN)    :: epsge
!!     Real(REAL64),                   Intent(IN)    :: maxstep
!!     Type(SISLIntcurve),             Intent(INOUT) :: intcurve
!!     Integer,                        Intent(IN)    :: makecurv
!!     Integer,                        Intent(IN)    :: graphic
!!     Integer,                        Intent(INOUT) :: stat


!---------------------------------- s1318 -------------------------------------

  Subroutine s1318(surf, centre, normal, cendist, radius, dim, epsco, epsge,   &
                   maxstep, intcurve, makecurv, graphic, stat)

!! PURPOSE
!!   s1318 - To march an intersection curve described by parameter pairs in an
!!           intersection curve object, a surface and a torus. The guide points
!!           are expected to be found by s1369. The generated geometric
!!          curves are represented as B-spline curves.

!! INTERFACE
!!   Subroutine s1318(surf, centre, normal, cendist, radius, dim, epsco,       &
!!                    epsge, maxstep, intcurve, makecurv, graphic, stat)
!!     Type(SISLsurf),                 Intent(IN)    :: surf
!!     Real(REAL64),                   Intent(IN)    :: centre(*)
!!     Real(REAL64),                   Intent(IN)    :: normal(*)
!!     Real(REAL64),                   Intent(IN)    :: cendist
!!     Real(REAL64),                   Intent(IN)    :: radius
!!     Integer,                        Intent(IN)    :: dim
!!     Real(REAL64),                   Intent(IN)    :: epsco
!!     Real(REAL64),                   Intent(IN)    :: epsge
!!     Real(REAL64),                   Intent(IN)    :: maxstep
!!     Type(SISLIntcurve),             Intent(INOUT) :: intcurve
!!     Integer,                        Intent(IN)    :: makecurv
!!     Integer,                        Intent(IN)    :: graphic
```

```
!!     Integer,                          Intent(INOUT) :: stat

!---------------------------------- s1310 -------------------------------------

  Subroutine s1310(surf1, surf2, intcurve, epsge, maxstep, makecurv, graphic,  &
                   stat)

!! PURPOSE
!!   s1310 - To march an intersection curve between two surfaces. The inter-
!!           section curve is described by guide parameter pairs stored in an
!!           intersection curve object. The guide points are expected to be
!!           found by s1859. The generated geometric curves are represented
!!           as B-spline curves.

!! INTERFACE
!!   Subroutine s1310(surf1, surf2, intcurve, epsge, maxstep, makecurv,        &
!!                    graphic, stat)
!!     Type(SISLsurf),     Intent(IN)    :: surf1
!!     Type(SISLsurf),     Intent(IN)    :: surf2
!!     Type(SISLIntcurve), Intent(INOUT) :: intcurve
!!     Real(REAL64),       Intent(IN)    :: epsge
!!     Real(REAL64),       Intent(IN)    :: maxstep
!!     Integer,            Intent(IN)    :: makecurv
!!     Integer,            Intent(IN)    :: graphic
!!     Integer,            Intent(INOUT) :: stat


!---------------------------------- s1319 -------------------------------------

  Subroutine s1319(surf, viewdir, dim, epsco, epsge, maxstep, intcurve,        &
                   makecurv, graphic, stat)

!! PURPOSE
!!   s1319 - To march the silhouette curve described by an intersection curve
!!           object, a surface and a view direction (i.e. parallel projection).
!!           The guide points are expected to be found by s1860. The generated
!!           geometric curves are represented as B-spline curves.

!! INTERFACE
!!   Subroutine s1319(surf, viewdir, dim, epsco, epsge, maxstep, intcurve,    &
!!                    makecurv, graphic, stat)
!!     Type(SISLsurf),     Intent(IN)    :: surf
!!     Real(REAL64),       Intent(IN)    :: viewdir(*)
!!     Integer,            Intent(IN)    :: dim
!!     Real(REAL64),       Intent(IN)    :: epsco
!!     Real(REAL64),       Intent(IN)    :: epsge
!!     Real(REAL64),       Intent(IN)    :: maxstep
!!     Type(SISLIntcurve), Intent(INOUT) :: intcurve
!!     Integer,            Intent(IN)    :: makecurv
!!     Integer,            Intent(IN)    :: graphic
!!     Integer,            Intent(INOUT) :: stat


!---------------------------------- s1514 -------------------------------------

  Subroutine s1514(ps1, eyepoint, idim, aepsco, aepsge, amax, pinter, icur,    &
                   igraph, jstat)

!! PURPOSE
```

```
!!   s1514 - To march the perspective silhouette curve described by an inter-
!!           section curve object, a surface and an eye point. The generated
!!           geometric curves are represented as B-spline curves.

!! INTERFACE
!!   Subroutine s1514(ps1, eyepoint, idim, aepsco, aepsge, amax, pinter,     &
!!                    icur, igraph, jstat)
!!     Type(SISLsurf),     Intent(IN)    :: ps1
!!     Real(REAL64),       Intent(IN)    :: eyepoint(*)
!!     Integer,            Intent(IN)    :: idim
!!     Real(REAL64),       Intent(IN)    :: aepsco
!!     Real(REAL64),       Intent(IN)    :: aepsge
!!     Real(REAL64),       Intent(IN)    :: amax
!!     Type(SISLIntcurve), Intent(INOUT) :: pinter
!!     Integer,            Intent(IN)    :: icur
!!     Integer,            Intent(IN)    :: igraph
!!     Integer,            Intent(INOUT) :: jstat


!---------------------------------- s1515 -------------------------------------

  Subroutine s1515(ps1, qpoint, bvec, idim, aepsco, aepsge, amax, pinter,      &
                   icur, igraph, jstat)

!! PURPOSE
!!   s1515 - To march the circular silhouette curve described by an intersection
!!           curve object, a surface, point Q and direction B i.e. solution of
!!           f(u, v) = N(u, v) × (P(u, v) − Q) · B.
!!           The generated geometric curves are represented as B-spline curves.

!! INTERFACE
!!   Subroutine s1515(ps1, qpoint, bvec, idim, aepsco, aepsge, amax, pinter,   &
!!                    icur, igraph, jstat)
!!     Type(SISLsurf),     Intent(IN)    :: ps1
!!     Real(REAL64),       Intent(IN)    :: qpoint(*)
!!     Real(REAL64),       Intent(IN)    :: bvec(*)
!!     Integer,            Intent(IN)    :: idim
!!     Real(REAL64),       Intent(IN)    :: aepsco
!!     Real(REAL64),       Intent(IN)    :: aepsge
!!     Real(REAL64),       Intent(IN)    :: amax
!!     Type(SISLIntcurve), Intent(INOUT) :: pinter
!!     Integer,            Intent(IN)    :: icur
!!     Integer,            Intent(IN)    :: igraph
!!     Integer,            Intent(INOUT) :: jstat


!---------------------------------- s1450 -------------------------------------

  Subroutine s1450(surf, epsge, close1, close2, degen1, degen2, degen3,        &
                   degen4, stat)

!! PURPOSE
!!  s1450 - To check if a surface is closed or has degenerate boundaries. The
!!          edge numbers are ordered counter-clockwise with edge no 1 at the
!!          v=0 parameteric location

!! INTERFACE
!!   Subroutine s1450(surf, epsge, close1, close2, degen1, degen2, degen3,    &
!!                    degen4, stat)
```

```
!!     Type(SISLsurf), Intent(IN)    :: surf
!!     Real(REAL64),   Intent(IN)    :: epsge
!!     Integer,        Intent(INOUT) :: close1
!!     Integer,        Intent(INOUT) :: close2
!!     Integer,        Intent(INOUT) :: degen1
!!     Integer,        Intent(INOUT) :: degen2
!!     Integer,        Intent(INOUT) :: degen3
!!     Integer,        Intent(INOUT) :: degen4
!!     Integer,        Intent(INOUT) :: stat


!---------------------------------- s1603 -------------------------------------

  Subroutine s1603(surf, min1, min2, max1, max2, stat)

!! PURPOSE
!!   s1603 - To pick the parameter ranges of a surface

!! INTERFACE
!!   Subroutine s1603(surf, min1, min2, max1, max2, stat)
!!     Type(SISLsurf),  Intent(IN)    :: surf
!!     Real(REAL64),    Intent(INOUT) :: min1
!!     Real(REAL64),    Intent(INOUT) :: min2
!!     Real(REAL64),    Intent(INOUT) :: max1
!!     Real(REAL64),    Intent(INOUT) :: max2
!!     Integer,         Intent(INOUT) :: stat


!---------------------------------- s1954 -------------------------------------

  Subroutine s1954(surf, point, dim, epsco, epsge, numclopt, pointpar,       &
                   numclocr, clocurves, stat)

!! PURPOSE
!!   s1954 - Find the points on a surface lying closest to a given point.

!! INTERFACE
!!   Subroutine s1954(surf, point, dim, epsco, epsge, numclopt, pointpar,   &
!!                    numclocr, clocurves, stat)
!!     Type(SISLsurf),                  Intent(IN)    :: surf
!!     Real(REAL64),                    Intent(IN)    :: point(*)
!!     Integer,                         Intent(IN)    :: dim
!!     Real(REAL64),                    Intent(IN)    :: epsco
!!     Real(REAL64),                    Intent(IN)    :: epsge
!!     Integer,                         Intent(INOUT) :: numclopt
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: pointpar(:)
!!     Integer,                         Intent(INOUT) :: numclocr
!!     Type(SISLIntcurve), ALLOCATABLE, Intent(INOUT) :: clocurves(:)
!!     Integer,                         Intent(INOUT) :: stat


!---------------------------------- s1958 -------------------------------------

  Subroutine s1958(psurf, epoint, idim, aepsco, aepsge, gpar, dist, jstat)

!! PURPOSE
!!   s1958 - Find the closest point between a surface and a point. The method
!!           is fast and should work well in clear cut cases, but there is no
!!           guarantee it will find the right solution. As long as it doesn't
```

```
!!          fail, it will find exactly one point. In other cases, use s1954

!! INTERFACE
!!   Subroutine s1958(psurf, epoint, idim, aepsco, aepsge, gpar, dist, jstat)
!!     Type(SISLsurf),     Intent(IN)    :: psurf
!!     Real(REAL64),       Intent(IN)    :: epoint(*)
!!     Integer,            Intent(IN)    :: idim
!!     Real(REAL64),       Intent(IN)    :: aepsco
!!     Real(REAL64),       Intent(IN)    :: aepsge
!!     Real(REAL64),       Intent(INOUT) :: gpar(2)
!!     Real(REAL64),       Intent(INOUT) :: dist
!!     Integer,            Intent(INOUT) :: jstat


!---------------------------------- s1775 -------------------------------------

  Subroutine s1775(surf, point, dim, epsge, start, end, guess, clpar, stat)

!! PURPOSE
!!   s1775 - Newton iteration on the distance function between a surface and
!!           a point, to find a closest point or an intersection point. If a bad
!!           choice for the guess parameters is given in, the iteration may end
!!           at a local, not global closest point.

!! INTERFACE
!!   Subroutine s1775(surf, point, dim, epsge, start, end, guess, clpar, stat)
!!     Type(SISLsurf),  Intent(IN)    :: surf
!!     Real(REAL64),    Intent(IN)    :: point(*)
!!     Integer,         Intent(IN)    :: dim
!!     Real(REAL64),    Intent(IN)    :: epsge
!!     Real(REAL64),    Intent(IN)    :: start(*)
!!     Real(REAL64),    Intent(IN)    :: end(*)
!!     Real(REAL64),    Intent(IN)    :: guess(*)
!!     Real(REAL64),    Intent(INOUT) :: clpar(*)
!!     Integer,         Intent(INOUT) :: stat


!---------------------------------- s1921 -------------------------------------

  Subroutine s1921(ps, edir, idim, aepsco, aepsge, jpt, gpar, jcrv, wcurve,    &
                   jstat)

!! PURPOSE
!!   s1921 - Find the absolute extremal points/curves of a surface along a given
!!           direction

!! INTERFACE
!!   Subroutine s1921(ps, edir, idim, aepsco, aepsge, jpt, gpar, jcrv,        &
!!                    wcurve, jstat)
!!     Type(SISLsurf),                  Intent(IN)    :: ps
!!     Real(REAL64),                    Intent(IN)    :: edir(*)
!!     Integer,                         Intent(IN)    :: idim
!!     Real(REAL64),                    Intent(IN)    :: aepsco
!!     Real(REAL64),                    Intent(IN)    :: aepsge
!!     Integer,                         Intent(INOUT) :: jpt
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: gpar(:)
!!     Integer,                         Intent(INOUT) :: jcrv
!!     Type(SISLIntcurve), ALLOCATABLE, Intent(INOUT) :: wcurve(:)
!!     Integer,                         Intent(INOUT) :: jstat
```

```
! newbox defined in Curve_Interrogation

!---------------------------------- s1989 -------------------------------------

  Subroutine s1989(ps, emax, emin, jstat)

!! PURPOSE
!!   s1989 - Find the bounding box of a surface.
!!           NOTE: The geometric bounding box is returned also in the ra-
!!           tional case, that is the box in homogeneous coordinates is NOT
!!           computed.

!! INTERFACE
!!   Subroutine s1989(ps, emax, emin, jstat)
!!     Type(SISLsurf),              Intent(IN)    :: ps
!!     Real(REAL64),   ALLOCATABLE, Intent(INOUT) :: emax(:)
!!     Real(REAL64),   ALLOCATABLE, Intent(INOUT) :: emin(:)
!!     Integer,                     Intent(INOUT) :: jstat

! newdir defined in Curve_Interrogation

!---------------------------------- s1987 -------------------------------------

  Subroutine s1987(ps, aepsge, jgtpi, gaxis, cang, jstat)

!! PURPOSE
!!   s1987 - Find the direction cone of a surface.

!! INTERFACE
!!   Subroutine s1987(ps, aepsge, jgtpi, gaxis, cang, jstat)
!!     Type(SISLsurf),             Intent(IN)    :: ps
!!     Real(REAL64),               Intent(IN)    :: aepsge
!!     Integer,                    Intent(INOUT) :: jgtpi
!!     Real(REAL64),  ALLOCATABLE, Intent(INOUT) :: gaxis(:)
!!     Real(REAL64),               Intent(INOUT) :: cang
!!     Integer,                    Intent(INOUT) :: jstat
```

B.8 Surface Analysis

```
!---------------------------------- s2500 -------------------------------------

  Subroutine s2500(surf, ider, iside1, iside2, parvalue, leftknot1,           &
                   leftknot2, gaussian, jstat)

!! PURPOSE
!!   s2500 - To compute the Gaussian curvature K(u,v) of a spline surface at
!!           given values (u,v) = (parvalue(1),parvalue(2)), where et1(leftknot1+1)
!!           <= parvalue(1) < et1(leftknot1+2) and et2[leftknot2+1] <= par-
!!           value(2) < et2([leftknot2+2). See also s2501().

!! INTERFACE
!!   Subroutine s2500(surf, ider, iside1, iside2, parvalue, leftknot1,      &
!!                    leftknot2, gaussian, jstat)
!!     Type(SISLsurf),  Intent(IN)    :: surf
!!     Integer,         Intent(IN)    :: ider
!!     Integer,         Intent(IN)    :: iside1
```

```
!!     Integer,         Intent(IN)    :: iside2
!!     Real(REAL64),    Intent(IN)    :: parvalue(*)
!!     Integer,         Intent(INOUT) :: leftknot1
!!     Integer,         Intent(INOUT) :: leftknot2
!!     Real(REAL64),    Intent(INOUT) :: gaussian
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s2502 -------------------------------------

  Subroutine s2502(surf, ider, iside1, iside2, parvalue, leftknot1,          &
                   leftknot2, meancurvature, jstat)

!! PURPOSE
!!   s2502 - To compute the mean curvature H(u,v) of a spline surface at given
!!           values (u,v) = (parvalue(1),parvalue(2)), where etl[leftknot1+1] <=
!!           parvalue(1) < etl[leftknot1+2] and et2[leftknot2+1] <= parvalue(2)
!!           < et2[leftknot2+2].

!! INTERFACE
!!   Subroutine s2502(surf, ider, iside1, iside2, parvalue, leftknot1,      &
!!                    leftknot2, meancurvature, jstat)
!!     Type(SISLsurf),  Intent(IN)    :: surf
!!     Integer,         Intent(IN)    :: ider
!!     Integer,         Intent(IN)    :: iside1
!!     Integer,         Intent(IN)    :: iside2
!!     Real(REAL64),    Intent(IN)    :: parvalue(*)
!!     Integer,         Intent(INOUT) :: leftknot1
!!     Integer,         Intent(INOUT) :: leftknot2
!!     Real(REAL64),    Intent(INOUT) :: meancurvature
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s2504 -------------------------------------

  Subroutine s2504(surf, ider, iside1, iside2, parvalue, leftknot1,          &
                   leftknot2, absCurvature, jstat)

!! PURPOSE
!!   s2504 - To compute the absolute curvature A(u,v) of a spline surface at
!!           given values (u,v) = (parvalue(1l,parvalue(2)), where et1[leftknot1+1]
!!           <= parvalue(1) < et1[leftknot1+2] and et2[leftknot2+1] <= par-
!!           value(2) < et2[leftknot2+2].

!! INTERFACE
!!   Subroutine s2504(surf, ider, iside1, iside2, parvalue, leftknot1,      &
!!                    leftknot2, absCurvature, jstat)
!!     Type(SISLsurf),  Intent(IN)    :: surf
!!     Integer,         Intent(IN)    :: ider
!!     Integer,         Intent(IN)    :: iside1
!!     Integer,         Intent(IN)    :: iside2
!!     Real(REAL64),    Intent(IN)    :: parvalue(*)
!!     Integer,         Intent(INOUT) :: leftknot1
!!     Integer,         Intent(INOUT) :: leftknot2
!!     Real(REAL64),    Intent(INOUT) :: absCurvature
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s2506 -------------------------------------

  Subroutine s2506(surf, ider, iside1, iside2, parvalue, leftknot1,          &
                   leftknot2, totalCurvature, jstat)
```

```
!! PURPOSE
!!   s2506 - To compute the total curvature T(u,v) of a surface at given val-
!!           ues (u,v) = (parvalue[1],parvalue[2]), where et1[leftknot1+1] <= par-
!!           value(1) < et1[leftknot1+2] and et2[leftknot2+1] <= parvalue(2) <
!!           et2[leftknot2+2].

!! INTERFACE
!!   Subroutine s2506(surf, ider, iside1, iside2, parvalue, leftknot1,        &
!!                    leftknot2, totalCurvature, jstat)
!!     Type(SISLsurf),  Intent(IN)    :: surf
!!     Integer,         Intent(IN)    :: ider
!!     Integer,         Intent(IN)    :: iside1
!!     Integer,         Intent(IN)    :: iside2
!!     Real(REAL64),    Intent(IN)    :: parvalue(*)
!!     Integer,         Intent(INOUT) :: leftknot1
!!     Integer,         Intent(INOUT) :: leftknot2
!!     Real(REAL64),    Intent(INOUT) :: totalCurvature
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s2508 -------------------------------------

  Subroutine s2508(surf, ider, iside1, iside2, parvalue, leftknot1,          &
                   leftknot2, mehlum, jstat)

!! PURPOSE
!!   s2508 - To compute the second order Mehlum curvature M(u,v) of
!!           a surface at given values (u,v) = (parvalue(1),parvalue(2)),
!!           where et1(leftknot1+1) <= parvalue(1) < et1(leftknot1+2) and
!!           et2(leftknot2+1) <= parvalue(2) < et2(leftknot2+2). See also s2509

!! INTERFACE
!!   Subroutine s2508(surf, ider, iside1, iside2, parvalue, leftknot1,         &
!!                    leftknot2, mehlum, jstat)
!!     Type(SISLsurf),  Intent(IN)    :: surf
!!     Integer,         Intent(IN)    :: ider
!!     Integer,         Intent(IN)    :: iside1
!!     Integer,         Intent(IN)    :: iside2
!!     Real(REAL64),    Intent(IN)    :: parvalue(*)
!!     Integer,         Intent(INOUT) :: leftknot1
!!     Integer,         Intent(INOUT) :: leftknot2
!!     Real(REAL64),    Intent(INOUT) :: mehlum
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s2510 -------------------------------------

  Subroutine s2510(surf, ider, iside1, iside2, parvalue, leftknot1,          &
                   leftknot2, mehlum, jstat)

!! PURPOSE
!!   s2510 - To compute the third order Mehlum curvature M(u,v) of a
!!           surface at given values (u,v) = (parvalue(1),parvalue(2)), where
!!           et1(leftknot1+1) <= parvalue(1) < et1(leftknot1+2), et2(leftknot2+1)
!!           <= parvalue(2) < et2(leftknot2+2).

!! INTERFACE
!!   Subroutine s2510(surf, ider, iside1, iside2, parvalue, leftknot1,      &
!!                    leftknot2, mehlum, jstat)
!!     Type(SISLsurf),  Intent(IN)    :: surf
```

```
!!     Integer,         Intent(IN)    :: ider
!!     Integer,         Intent(IN)    :: iside1
!!     Integer,         Intent(IN)    :: iside2
!!     Real(REAL64),    Intent(IN)    :: parvalue(*)
!!     Integer,         Intent(INOUT) :: leftknot1
!!     Integer,         Intent(INOUT) :: leftknot2
!!     Real(REAL64),    Intent(INOUT) :: mehlum
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s2532 -------------------------------------

  Subroutine s2532(surf, u_continuity, v_continuity, u_surfnumb, v_surfnumb,   &
                   gauss_surf, stat)

!! PURPOSE
!!   s2532 - To interpolate or approximate the Gaussian curvature of a B-spline
!!           or NURBS surface by a NURBS surface. The desired continuity
!!           of the Gaussian curvature surface is input and this may lead to a
!!           patchwork of output surfaces. Interpolation results in a high order
!!           surface. If the original surface is a B-spline surface of order k,
!!           the result is of order 8k − 11, in the NURBS case, order 32k − 35.
!!           To avoid instability because of this, a maximum order is applied.
!!           This may lead to an approximation rather than an interpolation.

!! INTERFACE
!!   Subroutine s2532(surf, u_continuity, v_continuity, u_surfnumb, v_surfnumb,&
!!                    gauss_surf, stat)
!!     Type(SISLsurf),              Intent(IN)    :: surf
!!     Integer,                     Intent(IN)    :: u_continuity
!!     Integer,                     Intent(IN)    :: v_continuity
!!     Integer,                     Intent(INOUT) :: u_surfnumb
!!     Integer,                     Intent(INOUT) :: v_surfnumb
!!     Type(SISLsurf), ALLOCATABLE, Intent(INOUT) :: gauss_surf(:)
!!     Integer,                     Intent(INOUT) :: stat

!---------------------------------- s2536 -------------------------------------

  Subroutine s2536(surf, u_continuity, v_continuity, u_surfnumb, v_surfnumb,   &
                   mehlum_surf, stat)

!! PURPOSE
!!   s2536 - To interpolate or approximate the Mehlum curvature of a B-spline
!!           or NURBS surface by a NURBS surface. The desired continuity
!!           of the Mehlum curvature surface is input and this may lead to a
!!           patchwork of output surfaces. Interpolation results in a high order
!!           surface. If the original surface is a B-spline surface of order k,
!!           the result is of order 12k − 17, in the NURBS case, order 48k − 53.
!!           To avoid instability beacuse of this, a maximum order is applied.
!!           This may lead to an approximation rather than an interpolation.

!! INTERFACE
!!   Subroutine s2536(surf, u_continuity, v_continuity, u_surfnumb, v_surfnumb,&
!!                    mehlum_surf, stat)
!!     Type(SISLsurf),              Intent(IN)    :: surf
!!     Integer,                     Intent(IN)    :: u_continuity
!!     Integer,                     Intent(IN)    :: v_continuity
!!     Integer,                     Intent(INOUT) :: u_surfnumb
!!     Integer,                     Intent(INOUT) :: v_surfnumb
!!     Type(SISLsurf), ALLOCATABLE, Intent(INOUT) :: mehlum_surf(:)
```

```
!!     Integer,                     Intent(INOUT) :: stat

!---------------------------------- s2540 -------------------------------------

  Subroutine s2540(surf, curvature_type, export_par_val, pick_subpart,         &
                   boundary, n_u, n_v, garr, stat)

!! PURPOSE
!!   s2540 - To compute a set of curvature values on a uniform grid in a
!!           selected subset of the parameter domain of a NURBS surface.

!! INTERFACE
!!   Subroutine s2540(surf, curvature_type, export_par_val, pick_subpart,      &
!!                    boundary, n_u, n_v, garr, stat)
!!     Type(SISLsurf),                  Intent(IN)    :: surf
!!     Integer,                         Intent(IN)    :: curvature_type
!!     Integer,                         Intent(IN)    :: export_par_val
!!     Integer,                         Intent(IN)    :: pick_subpart
!!     Real(REAL64),                    Intent(IN)    :: boundary(*)
!!     Integer,                         Intent(IN)    :: n_u
!!     Integer,                         Intent(IN)    :: n_v
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: garr(:)
!!     Integer,                         Intent(INOUT) :: stat

!---------------------------------- s2542 -------------------------------------

  Subroutine s2542(surf, ider, iside1, iside2, parvalue, leftknot1,          &
                   leftknot2, k1, k2, d1, d2, jstat)

!! PURPOSE
!!   s2542 - To compute principal curvatures (k1,k2) with corresponding
!!           principal directions (d1,d2) of a spline surface at given values
!!           (u,v) = (parvalue(1),parvalue(2)), where etl[leftknot1+1] <= par-
!!           value(1) < etl[leftknot1+2] and et2[leftknot2+1] <= parvalue(2) <
!!           et2(leftknot2+2).

!! INTERFACE
!!   Subroutine s2542(surf, ider, iside1, iside2, parvalue, leftknot1,      &
!!                    leftknot2, k1, k2, d1, d2, jstat)
!!     Type(SISLsurf),  Intent(IN)    :: surf
!!     Integer,         Intent(IN)    :: ider
!!     Integer,         Intent(IN)    :: iside1
!!     Integer,         Intent(IN)    :: iside2
!!     Real(REAL64),    Intent(IN)    :: parvalue(*)
!!     Integer,         Intent(INOUT) :: leftknot1
!!     Integer,         Intent(INOUT) :: leftknot2
!!     Real(REAL64),    Intent(INOUT) :: k1
!!     Real(REAL64),    Intent(INOUT) :: k2
!!     Real(REAL64),    Intent(INOUT) :: d1(*)
!!     Real(REAL64),    Intent(INOUT) :: d2(*)
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s2544 -------------------------------------

  Subroutine s2544(surf, ider, iside1, iside2, parvalue, leftknot1,          &
                   leftknot2, norcurv, jstat)

!! PURPOSE
!!   s2544 - To compute the Normal curvature of a splne surface at
```

```
!!           given values (u,v) = (parvalue(1),parvalue[2)) in the direc-
!!           tion (parvalue(2),parvalue(l)) where et1(leftknot1+1) <= par-
!!           value(1) < et1(leftknot1+2) and et2(leftknot2+1) <= parvalue(2) <
!!           et2(leftknot2+2).

!! INTERFACE
!!   Subroutine s2544(surf, ider, iside1, iside2, parvalue, leftknot1,      &
!!                    leftknot2, norcurv, jstat)
!!     Type(SISLsurf),  Intent(IN)    :: surf
!!     Integer,         Intent(IN)    :: ider
!!     Integer,         Intent(IN)    :: iside1
!!     Integer,         Intent(IN)    :: iside2
!!     Real(REAL64),    Intent(IN)    :: parvalue(*)
!!     Integer,         Intent(INOUT) :: leftknot1
!!     Integer,         Intent(INOUT) :: leftknot2
!!     Real(REAL64),    Intent(INOUT) :: norcurv(*)
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s2545 -------------------------------------

  Subroutine s2545(surf, curvature_type, export_par_val, boundary, n_u, n_v,   &
                   scale, garr, stat)

!! PURPOSE
!!   s2545 - To compute a set of focal values on a uniform grid in a selected
!!           subset of the parameter domain of a NURBS surface. A focal
!!           value is a surface position offset by the surface curvature.

!! INTERFACE
!!   Subroutine s2545(surf, curvature_type, export_par_val, boundary, n_u, n_v,&
!!                    scale, garr, stat)
!!     Type(SISLsurf),                  Intent(IN)    :: surf
!!     Integer,                         Intent(IN)    :: curvature_type
!!     Integer,                         Intent(IN)    :: export_par_val
!!     Real(REAL64),                    Intent(IN)    :: boundary(*)
!!     Integer,                         Intent(IN)    :: n_u
!!     Integer,                         Intent(IN)    :: n_v
!!     Real(REAL64),                    Intent(IN)    :: scale
!!     Real(REAL64),       ALLOCATABLE, Intent(INOUT) :: garr(:)
!!     Integer,                         Intent(INOUT) :: stat
```

B.9 Surface Utilities

```
!---------------------------------- newSurf -----------------------------------

  Subroutine newSurf(number1, number2, order1, order2, knot1, knot2, coef,     &
                   kind, dim, copy, surf)

!! PURPOSE
!!   newSurf - Create and initialize a surface object instance.

!! INTERFACE
!!   Subroutine newSurf(number1, number2, order1, order2, knot1, knot2, coef, &
!!                      kind, dim, copy, surf)
!!     Integer,        Intent(IN)    :: number1
!!     Integer,        Intent(IN)    :: number2
!!     Integer,        Intent(IN)    :: order1
```

```
!!     Integer,        Intent(IN)    :: order2
!!     Integer,        Intent(IN)    :: kind
!!     Integer,        Intent(IN)    :: dim
!!     Integer,        Intent(IN)    :: copy
!!     Real(REAL64),   Intent(IN)    :: knot1(*)
!!     Real(REAL64),   Intent(IN)    :: knot2(*)
!!     Real(REAL64),   Intent(IN)    :: coef(*)
!!     Type(SISLsurf), Intent(INOUT) :: surf

!-------------------------------- copySurface --------------------------------

  Subroutine copySurface(old_surf, new_surf)

!! PURPOSE
!!   copySurface - Make a copy of a SISLSurface object.

!! INTERFACE
!!   Subroutine copySurface(old_surf, new_surf)
!!     Type(SISLSurf), Intent(IN)    :: old_surf
!!     Type(SISLSurf), Intent(INOUT) :: new_surf

!-------------------------------- freeSurf ----------------------------------

  Subroutine freeSurf(surf)

!! PURPOSE
!!   freeSurf - Free the space occupied by the surface. Before using freeSurf,
!!              make sure that the surface object exists.

!! INTERFACE
!!   Subroutine freeSurf(surf)
!!     Type(SISLSurf), Intent(INOUT) :: surf

!-------------------------------- s1421 -------------------------------------

  Subroutine s1421(surf, der, parvalue, leftknot1, leftknot2, derive, normal,  &
                   stat)

!! PURPOSE
!!   s1421 - Evaluate the surface at a given parameter value pair. Compute
!!           der derivatives and the normal if der ≥ 1. See also s1424

!! INTERfACE
!!   Subroutine s1421(surf, der, parvalue, leftknot1, leftknot2, derive,       &
!!                    normal, stat)
!!     Type(SISLsurf),  Intent(IN)    :: surf
!!     Integer,         Intent(IN)    :: der
!!     Real(REAL64),    Intent(IN)    :: parvalue(*)
!!     Integer,         Intent(INOUT) :: leftknot1
!!     Integer,         Intent(INOUT) :: leftknot2
!!     Real(REAL64),    Intent(INOUT) :: derive(*)
!!     Real(REAL64),    Intent(INOUT) :: normal(*)
!!     Integer,         Intent(INOUT) :: stat


!-------------------------------- s1424 -------------------------------------

  Subroutine s1424(surf, der1, der2, parvalue, leftknot1, leftknot2, derive,   &
                   stat)
```

```
!! PURPOSE
!!   s1424 - Evaluate the surface the parameter value (parvalue(1), par-
!!           value(2)). Compute the der1 × der2 first derivatives. The deriva-
!!           tives that will be computed are D i,j , i = 1, 2, . . . , der1+1,
!!           j=1, 2, ..., der2+1.

!! INTERFACE
!!   Subroutine s1424(surf, der1, der2, parvalue, leftknot1, leftknot2,   &
!!                    derive, stat)
!!     Type(SISLsurf),  Intent(IN)    :: surf
!!     Integer,         Intent(IN)    :: der1
!!     Integer,         Intent(IN)    :: der2
!!     Real(REAL64),    Intent(IN)    :: parvalue(*)
!!     Integer,         Intent(INOUT) :: leftknot1
!!     Integer,         Intent(INOUT) :: leftknot2
!!     Real(REAL64),    Intent(INOUT) :: derive(*)
!!     Integer,         Intent(INOUT) :: stat

!---------------------------------- s1422 -------------------------------------

  Subroutine s1422(ps1, ider, iside1, iside2, epar, ilfs, ilft, eder, enorm,  &
                   jstat)

!! PURPOSE
!!   s1422 - Evaluate and compute the left- or right-hand derivatives of a sur-
!!           face at a given parameter position.

!! INTERFACE
!!   Subroutine s1422(ps1, ider, iside1, iside2, epar, ilfs, ilft, eder,   &
!!                    enorm, jstat)
!!     Type(SISLsurf),  Intent(IN)    :: ps1
!!     Integer,         Intent(IN)    :: ider
!!     Integer,         Intent(IN)    :: iside1
!!     Integer,         Intent(IN)    :: iside2
!!     Real(REAL64),    Intent(IN)    :: epar(*)
!!     Integer,         Intent(INOUT) :: ilfs
!!     Integer,         Intent(INOUT) :: ilft
!!     Real(REAL64),    Intent(INOUT) :: eder(*)
!!     Real(REAL64),    Intent(INOUT) :: enorm(*)
!!     Integer,         Intent(INOUT) :: jstat


!---------------------------------- s1425 -------------------------------------

  Subroutine s1425(ps1, ider1, ider2, iside1, iside2, epar, ileft1, ileft2,    &
                   eder, jstat)

!! PURPOSE
!!   s1425 - To compute the value and ider1 × ider2 first derivatives of a ten-
!!           sor product surface at the point with parameter value (epar(1),
!!           epar(2)). The derivatives that will be computed are D(i, j),
!!           i = 1, 2, . . . , ider1+1, j = 1, 2, . . . , ider2+1. The
!!           calculations are from the right hand or left hand side.

!! INTERFACE
!!   Subroutine s1425(ps1, ider1, ider2, iside1, iside2, epar, ileft1, ileft2, &
!!                    eder, jstat)
!!     Type(SISLsurf),  Intent(IN)    :: ps1
```

```
!!     Integer,         Intent(IN)    :: ider1
!!     Integer,         Intent(IN)    :: ider2
!!     Integer,         Intent(IN)    :: iside1
!!     Integer,         Intent(IN)    :: iside2
!!     Real(REAL64),    Intent(IN)    :: epar(*)
!!     Integer,         Intent(INOUT) :: ileft1
!!     Integer,         Intent(INOUT) :: ileft2
!!     Real(REAL64),    Intent(INOUT) :: eder(*)
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s1506 -------------------------------------

  Subroutine s1506(ps1, ider, m1, x, m2, y, eder, norm, jstat)

!! PURPOSE
!!   s1506 - Evaluate the surface pointed at by ps1 over an m1 * m2 grid of
!!           points (x(i),y(j)). Compute ider derivatives and normals if
!!           suitable.

!! INTERFACE
!!   Subroutine s1506(ps1, ider, m1, x, m2, y, eder, norm, jstat)
!!     Type(SISLsurf),  Intent(IN)    :: ps1
!!     Integer,         Intent(IN)    :: ider
!!     Integer,         Intent(IN)    :: m1
!!     Real(REAL64),    Intent(IN)    :: x(*)
!!     Integer,         Intent(IN)    :: m2
!!     Real(REAL64),    Intent(IN)    :: y(*)
!!     Real(REAL64),    Intent(INOUT) :: eder(*)
!!     Real(REAL64),    Intent(INOUT) :: norm(*)
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s1711 -------------------------------------

  Subroutine s1711(surf, pardir, parval, newsurf1, newsurf2, stat)

!! PURPOSE
!!   s1711 - Subdivide a surface along a given internal parameter line.

!! INTERFACE
!!   Subroutine s1711(surf, pardir, parval, newsurf1, newsurf2, stat)
!!     Type(SISLsurf),  Intent(IN)    :: surf
!!     Integer,         Intent(IN)    :: pardir
!!     Real(REAL64),    Intent(IN)    :: parval
!!     Type(SISLsurf),  Intent(INOUT) :: newsurf1
!!     Type(SISLsurf),  Intent(INOUT) :: newsurf2
!!     Integer,         Intent(INOUT) :: stat


!---------------------------------- s1025 -------------------------------------

  Subroutine s1025(ps, epar1, inpar1, epar2, inpar2, rsnew, jstat)

!! PURPOSE
!!   s1025 - Insert a given set of knots in each parameter direction into the
!!           description of a surface.
!!           NOTE : When the surface is periodic in one direction, the input
!!           parameter values in this direction must lie in the half-open
!!           interval (et(kk), et(kn+1)), the function will automatically update
!!           the extra knots and coefficients.
```

```
!! INTERFACE
!!   Subroutine s1025(ps, epar1, inpar1, epar2, inpar2, rsnew, jstat)
!!     Type(SISLsurf),  Intent(IN)    :: ps
!!     Real(REAL64),    Intent(IN)    :: epar1(*)
!!     Integer,         Intent(IN)    :: inpar1
!!     Real(REAL64),    Intent(IN)    :: epar2(*)
!!     Integer,         Intent(IN)    :: inpar2
!!     Type(SISLsurf),  Intent(INOUT) :: rsnew
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s1439 -------------------------------------

  Subroutine s1439(ps1, apar, idirec, rcurve, jstat)

!! PURPOSE
!!   s1439 - Make a constant parameter curve along a given parameter direc-
!!           tion in a surface.

!! INTERFACE
!!   Subroutine s1439(ps1, apar, idirec, rcurve, jstat)
!!     Type(SISLsurf),  Intent(IN)    :: ps1
!!     Real(REAL64),    Intent(IN)    :: apar
!!     Integer,         Intent(IN)    :: idirec
!!     Type(SISLcurve), Intent(INOUT) :: rcurve
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s1383 -------------------------------------

  Subroutine s1383(surf, curve, epsge, maxstep, der, newcurve1, newcurve2,     &
                   newcurve3, stat)

!! PURPOSE
!!   s1383 - To create a 3D approximation to the curve in a surface, traced
!!           out by a curve in the parameter plane. The output is represented
!!           as a B-spline curve.

!! INTERFACE
!!   Subroutine s1383(surf, curve, epsge, maxstep, der, newcurve1, newcurve2, &
!!                    newcurve3, stat)
!!     Type(SISLsurf),  Intent(IN)    :: surf
!!     Type(SISLcurve), Intent(IN)    :: curve
!!     Real(REAL64),    Intent(IN)    :: epsge
!!     Real(REAL64),    Intent(IN)    :: maxstep
!!     Integer,         Intent(IN)    :: der
!!     Type(SISLcurve), Intent(INOUT) :: newcurve1
!!     Type(SISLcurve), Intent(INOUT) :: newcurve2
!!     Type(SISLcurve), Intent(INOUT) :: newcurve3
!!     Integer,         Intent(INOUT) :: stat


!---------------------------------- s1001 -------------------------------------

  Subroutine s1001(ps, min1, min2, max1, max2, rsnew, jstat)

!! PURPOSE
!!   s1001 - To pick a part of a surface. The surface produced will always be
!!           k-regular, i.e. with k-tupple start/end knots.
```

```
!! INTERFACE
!!   Subroutine s1001(ps, min1, min2, max1, max2, rsnew, jstat)
!!     Type(SISLsurf),  Intent(IN)    :: ps
!!     Real(REAL64),    Intent(IN)    :: min1
!!     Real(REAL64),    Intent(IN)    :: min2
!!     Real(REAL64),    Intent(IN)    :: max1
!!     Real(REAL64),    Intent(IN)    :: max2
!!     Type(SISLsurf),  Intent(INOUT) :: rsnew
!!     Integer,         Intent(INOUT) :: jstat


!---------------------------------- s1440 -------------------------------------

  Subroutine s1440(surf, newsurf, stat)

!! PURPOSE
!!   s1440 - Interchange the two parameter directions used in the mathemat-
!!           ical description of a surface and thereby change the direction of
!!           the normal vector of the surface.

!! INTERFACE
!!   Subroutine s1440(surf, newsurf, stat)
!!     Type(SISLsurf),  Intent(IN)    :: surf
!!     Type(SISLsurf),  Intent(INOUT) :: newsurf
!!     Integer,         Intent(INOUT) :: stat
```

B.10 Data Reduction

```
!---------------------------------- s1940 -------------------------------------

  Subroutine s1940(oldcurve, eps, startfix, endfix, iopen, itmax, newcurve,    &
                   maxerr, stat)

!! PURPOSE
!!   s1940 - To remove as many knots as possible from a spline curve without
!!           perturbing the curve more than a given tolerance.

!! INTERFACE
!!   Subroutine s1940(oldcurve, eps, startfix, endfix, iopen, itmax, newcurve, &
!!                    maxerr, stat)
!!     Type(SISLcurve), Intent(IN)    :: oldcurve
!!     Real(REAL64),    Intent(IN)    :: eps(*)
!!     Integer,         Intent(IN)    :: startfix
!!     Integer,         Intent(IN)    :: endfix
!!     Integer,         Intent(IN)    :: iopen
!!     Integer,         Intent(IN)    :: itmax
!!     Type(SISLcurve), Intent(INOUT) :: newcurve
!!     Real(REAL64),    Intent(INOUT) :: maxerr(*)
!!     Integer,         Intent(INOUT) :: stat


!---------------------------------- s1961 -------------------------------------

  Subroutine s1961(ep, im, idim, ipar, epar, eeps, ilend, irend, iopen,        &
                   afctol, itmax, ik, rc, emxerr, jstat)

!! PURPOSE
```

```
!!   s1961 - To compute a spline-approximation to the data given by the points
!!           ep, and represent it as a B-spline curve with parameterization de-
!!           termined by the parameter ipar. The approximation is determined
!!           by first forming the piecewise linear interpolant to the data, and
!!           then performing knot removal on this initial approximation.

!! INTERFACE
!!   Subroutine s1961(ep, im, idim, ipar, epar, eeps, ilend, irend, iopen,     &
!!                    afctol, itmax, ik, rc, emxerr, jstat)
!!     Integer,         Intent(IN)    :: im
!!     Integer,         Intent(IN)    :: idim
!!     Integer,         Intent(IN)    :: ipar
!!     Real(REAL64),    Intent(IN)    :: epar(:)
!!     Real(REAL64),    Intent(IN)    :: eeps(*)
!!     Integer,         Intent(IN)    :: ilend
!!     Integer,         Intent(IN)    :: irend
!!     Integer,         Intent(IN)    :: iopen
!!     Real(REAL64),    Intent(IN)    :: afctol
!!     Integer,         Intent(IN)    :: itmax
!!     Integer,         Intent(IN)    :: ik
!!     Type(SISLcurve), Intent(INOUT) :: rc
!!     Real(REAL64),    Intent(INOUT) :: emxerr(*)
!!     Integer,         Intent(INOUT) :: jstat


!---------------------------------- s1962 -------------------------------------

  Subroutine s1962(ep, ev, im, idim, ipar, epar, eeps, ilend, irend, iopen,    &
                   itmax, rc, emxerr, jstat)

!! PURPOSE
!!   s1962 - To compute the approximation to the data given by the points
!!           ep and the derivatives (tangents) ev, and represent it as a B-
!!           spline curve with parametrization determined by the parameter
!!           ipar. The approximation is determined by first forming the cubic
!!           hermite interpolant to the data, and then performing knot removal
!!           on this initial approximation.

!! INTERFACE
!!   Subroutine s1962(ep, ev, im, idim, ipar, epar, eeps, ilend, irend, iopen,    &
!!                    itmax, rc, emxerr, jstat)
!!     Real(REAL64),    Intent(IN)    :: ep(*)
!!     Real(REAL64),    Intent(IN)    :: ev(*)
!!     Integer,         Intent(IN)    :: im
!!     Integer,         Intent(IN)    :: idim
!!     Integer,         Intent(IN)    :: ipar
!!     Real(REAL64),    Intent(IN)    :: epar(:)
!!     Real(REAL64),    Intent(IN)    :: eeps(*)
!!     Integer,         Intent(IN)    :: ilend
!!     Integer,         Intent(IN)    :: irend
!!     Integer,         Intent(IN)    :: iopen
!!     Integer,         Intent(IN)    :: itmax
!!     Type(SISLcurve), Intent(INOUT) :: rc
!!     Real(REAL64),    Intent(INOUT) :: emxerr(*)
!!     Integer,         Intent(INOUT) :: jstat

!---------------------------------- s1963 -------------------------------------

  Subroutine s1963(pc, eeps, ilend, irend, iopen, itmax, rc, emxerr, jstat)
```

```
!! PURPOSE
!!   s1963 - To approximate the input spline curve by a cubic spline curve with
!!           error less than eeps in each of the kdim components.

!! INTERFACE
!!   Subroutine s1963(pc, eeps, ilend, irend, iopen, itmax, rc, emxerr, jstat)
!!     Type(SISLcurve), Intent(IN)    :: pc
!!     Real(REAL64),    Intent(IN)    :: eeps(*)
!!     Integer,         Intent(IN)    :: ilend
!!     Integer,         Intent(IN)    :: irend
!!     Integer,         Intent(IN)    :: iopen
!!     Integer,         Intent(IN)    :: itmax
!!     Type(SISLcurve), Intent(INOUT) :: rc
!!     Real(REAL64),    Intent(INOUT) :: emxerr(*)
!!     Integer,         Intent(INOUT) :: jstat


!---------------------------------- s1965 -------------------------------------

  Subroutine s1965(oldsurf, eps, edgefix, iopen1, iopen2, edgeps, opt, itmax,  &
                   newsurf, maxerr, stat)

!! PURPOSE
!!   s1965 - To remove as many knots as possible from a spline surface without
!!           perturbing the surface more than the given tolerance. The error
!!           in continuity over the start and end of a closed or periodic surface
!!           is only guaranteed to be within edgeps.

!! INTERFACE
!!   Subroutine s1965(oldsurf, eps, edgefix, iopen1, iopen2, edgeps, opt,   &
!!                    itmax, newsurf, maxerr, stat)
!!     Type(SISLsurf),  Intent(IN)    :: oldsurf
!!     Real(REAL64),    Intent(IN)    :: eps(*)
!!     Integer,         Intent(IN)    :: edgefix(4)
!!     Integer,         Intent(IN)    :: iopen1
!!     Integer,         Intent(IN)    :: iopen2
!!     Real(REAL64),    Intent(IN)    :: edgeps(*)
!!     Integer,         Intent(IN)    :: opt
!!     Integer,         Intent(IN)    :: itmax
!!     Type(SISLsurf),  Intent(INOUT) :: newsurf
!!     Real(REAL64),    Intent(INOUT) :: maxerr(*)
!!     Integer,         Intent(INOUT) :: stat


!---------------------------------- s1966 -------------------------------------

  Subroutine s1966(ep, im1, im2, idim, ipar, epar1, epar2, eeps, nend, iopen1, &
                   iopen2, edgeps, afctol, iopt, itmax, ik1, ik2, rs, emxerr,  &
                   jstat)

!! PURPOSE
!!   s1966 - To compute a tensor-product spline-approximation of order
!!           (ik1,ik2) to the rectangular array of idim-dimensional points given
!!           by ep.

!! INTERFACE
!!   Subroutine s1966(ep, im1, im2, idim, ipar, epar1, epar2, eeps, nend,      &
!!                    iopen1, iopen2, edgeps, afctol, iopt, itmax, ik1, ik2,   &
```

```
!!                     rs, emxerr, jstat)
!!     Real(REAL64),    Intent(IN)    :: ep(*)
!!     Integer,         Intent(IN)    :: im1
!!     Integer,         Intent(IN)    :: im2
!!     Integer,         Intent(IN)    :: idim
!!     Integer,         Intent(IN)    :: ipar
!!     Real(REAL64),    Intent(IN)    :: epar1(:)
!!     Real(REAL64),    Intent(IN)    :: epar2(:)
!!     Real(REAL64),    Intent(IN)    :: eeps(*)
!!     Integer,         Intent(IN)    :: nend(4)
!!     Integer,         Intent(IN)    :: iopen1
!!     Integer,         Intent(IN)    :: iopen2
!!     Real(REAL64),    Intent(IN)    :: edgeps(*)
!!     Real(REAL64),    Intent(IN)    :: afctol
!!     Integer,         Intent(IN)    :: iopt
!!     Integer,         Intent(IN)    :: itmax
!!     Integer,         Intent(IN)    :: ik1
!!     Integer,         Intent(IN)    :: ik2
!!     Type(SISLsurf),  Intent(INOUT) :: rs
!!     Real(REAL64),    Intent(INOUT) :: emxerr(*)
!!     Integer,         Intent(INOUT) :: jstat


!---------------------------------- s1967 -------------------------------------

  Subroutine s1967(ep, etang1, etang2, eder11, im1, im2, idim, ipar, epar1,    &
                   epar2, eeps, nend, iopen1, iopen2, edgeps, iopt, itmax, rs, &
                   emxerr, jstat)

!! PURPOSE
!!   s1967 - To compute a bicubic hermite spline-approximation to the position
!!           and derivative data given by ep,etang1,etang2 and eder11.

!! INTERFACE
!!   Subroutine s1967(ep, etang1, etang2, eder11, im1, im2, idim, ipar,        &
!!                    epar1, epar2, eeps, nend, iopen1, iopen2, edgeps, iopt,  &
!!                    itmax, rs, emxerr, jstat)
!!     Real(REAL64),    Intent(IN)    :: ep(*)
!!     Real(REAL64),    Intent(IN)    :: etang1(*)
!!     Real(REAL64),    Intent(IN)    :: etang2(*)
!!     Real(REAL64),    Intent(IN)    :: eder11(*)
!!     Integer,         Intent(IN)    :: im1
!!     Integer,         Intent(IN)    :: im2
!!     Integer,         Intent(IN)    :: idim
!!     Integer,         Intent(IN)    :: ipar
!!     Real(REAL64),    Intent(IN)    :: epar1(:)
!!     Real(REAL64),    Intent(IN)    :: epar2(:)
!!     Real(REAL64),    Intent(IN)    :: eeps(*)
!!     Integer,         Intent(IN)    :: nend(4)
!!     Integer,         Intent(IN)    :: iopen1
!!     Integer,         Intent(IN)    :: iopen2
!!     Real(REAL64),    Intent(IN)    :: edgeps(*)
!!     Integer,         Intent(IN)    :: iopt
!!     Integer,         Intent(IN)    :: itmax
!!     Type(SISLsurf),  Intent(INOUT) :: rs
!!     Real(REAL64),    Intent(INOUT) :: emxerr(*)
!!     Integer,         Intent(INOUT) :: jstat
```

```
!---------------------------------- s1968 -------------------------------------

  Subroutine s1968(ps, eeps, nend, iopen1, iopen2, edgeps, iopt, itmax, rs,    &
                   jstat)

!! PURPOSE
!!   s1968 - To compute a cubic tensor-product spline approximation to a
!!           given tensor product spline surface of arbitrary order, with er-
!!           ror less than eeps in each of the idim components. The error in
!!           continuity over the start and end of a closed or periodic surface
!!           is only guaranteed to be within edgeps.

!! INTERFACE
!!   Subroutine s1968(ps, eeps, nend, iopen1, iopen2, edgeps, iopt, itmax, rs,&
!!                    jstat)
!!     Type(SISLsurf),  Intent(IN)    :: ps
!!     Real(REAL64),    Intent(IN)    :: eeps(*)
!!     Integer,         Intent(IN)    :: nend(4)
!!     Integer,         Intent(IN)    :: iopen1
!!     Integer,         Intent(IN)    :: iopen2
!!     Real(REAL64),    Intent(IN)    :: edgeps(*)
!!     Integer,         Intent(IN)    :: iopt
!!     Integer,         Intent(IN)    :: itmax
!!     Type(SISLsurf),  Intent(INOUT) :: rs
!!     Integer,         Intent(INOUT) :: jstat
```

B.11 forSISL IO

```
  Function determine_SISL_instance_type(is) RESULT(itype)

!! PURPOSE
!!   Reads header record of input files to determine in file contains the
!!   appropriate data type for curve or surface io

!! INTERFACE
!!   Function determine_SISL_instance_type(is) RESULT(itype)
!!     Integer, Intent(IN) :: is
!!     Integer             :: itype


  Subroutine read_basis(is, n, k, knots)

!! PURPOSE
!!   Reads knot vector information for curve or surface objects. All io is done
!!   using Fortran list directed IO

!! INTERFACE
!!   Subroutine read_basis(is, n, k, knots)
!!     Integer,                    Intent(IN)    :: is
!!     Integer,                    Intent(INOUT) :: n, k
!!     Real(REAL64),  ALLOCATABLE, Intent(INOUT) :: knots(:)


  Subroutine write_basis(os, n, k, knots)

!! PURPOSE
!!   Writes knot vector data for curve and surface objects. Integers are
```

```
!!   written using "i0,' ',i0" format. Reals are written using "500(g0.7,' ')"
!!   format

!! INTERFACE
!!   Subroutine write_basis(os, n, k, knots)
!!     Integer,        Intent(IN) :: os
!!     Integer,        Intent(IN) :: n, k
!!     Real(REAL64),   Intent(IN) :: knots(:)


  Subroutine readSISLSurface(SISL_file, surf)

!! PURPOSE
!!   Read SISL surface object data

!! INTERFACE
!!   Subroutine readSISLSurface(SISL_file, surf)
!!     Integer,        Intent(IN)    :: SISL_file
!!     Type(SISLsurf), Intent(INOUT) :: surf


  Subroutine writeSISLSurface(surf, SISL_file)

!! PURPOSE
!!   Write SISL surface object data

!! INTERFACE
!!   Subroutine writeSISLSurface(surf, SISL_file)
!!     Type(SISLsurf), Intent(IN) :: surf
!!     Integer,        Intent(IN) :: SISL_file


  Subroutine writeSISLCurve(curve, SISL_file)

!! PURPOSE
!!   Write SISL curve object data

!! INTERFACE
!!   Subroutine writeSISLCurve(curve, SISL_file)
!!     Type(SISLcurve), Intent(IN) :: curve
!!     Integer,         Intent(IN) :: SISL_file


  Subroutine readSISLCurve(SISL_file, curve)

!! PURPOSE
!!   Read SISL curve object data

!! INTERFACE
!!   Subroutine readSISLcurve(SISL_file, curve)
!!     Integer,         Intent(IN)     :: SISL_file
!!     Type(SISLcurve), Intent(INOUT) :: curve

  Subroutine writeSISLPoints(num_points, coords, SISL_file)

!! PURPOSE
!!   Write point data generated by SISL routines

!! INTERFACE
```

```
!!   Subroutine WriteSISLPoints(SISL_file, coords)
!!     Integer,      Intent(IN)         :: num_points
!!     Real(REAL64), TARGET, Intent(IN) :: coords(:)
!!     Integer,              Intent(IN) :: SISL_file


  Subroutine readSISLPoints(coords, SISL_file)

!! PURPOSE
!!   Read point data generated by SISL routines

!! INTERFACE
!!   Subroutine readSISLPoints(coords, SISL_file)
!!     Real(REAL64), ALLOCATABLE, Intent(INOUT) :: coords(:)
!!     Integer,                   Intent(IN)    :: SISL_file
```