BsplineEval

A C-library to evaluate a spline from its B-spline representation.
edition 1.1.7 for `BsplineEval` version 1.1.7
17 May 2013

Dan Nguyen

Table of Contents

1 Copying

2 Introduction

`BsplineEval` This library implements the algorithms to evaluate a spline from its B-spline representation as described in the paper:

The Numerical Evaluation of a Spline from its B-Spline Representation by M. G. Cox. J. Inst. Maths. Applics (1978) 21, 135-143.

The `Algorithm A`, `Algorithm B` and `Algorithm C` as described in said paper are implemented. Moreover, the `algorithm C` with its associated derivatives and an optimized version of algorithm C with its derivatives are provided.

The functions `Bspline_Algo_a` and `Bspline_Algo_b` are the one dimensional spline evaluation from its B-spline representation using algorithm A and B, respectively.

The evaluation of s(theta, z), where theta is periodic, from its normalized B-spline representation are implemented in the functions: `Bspline_Algo_c`, `Bspline_Algo_c_derivs`, `Bspline_eval` and `Bspline_eval_derivs`.

The functions `Bspline_Algo_c` and `Bspline_Algo_c_derivs` allow the user to set the underlying algorithms used (algorithm 'a' or 'b'). `Bspline_eval` and `Bspline_eval_derivs` are the optimized versions of `Bspline_Algo_c` and `Bspline_Algo_c_derivs`, respectively.

Functions with a *_derivs* suffix also evaluates the derivatives with respect to theta and z. The results of the B-spline evaluation and its derivatives are stored in the structure:

```
typedef struct {

  double s                                              /* s( theta, z ) */;
  double dsdz                                       /* ds( theta, z ) / dz */;
  double dsdt                                           /* ds( theta, t ) */;

} BsplineResult;
```

3 Example

3.1 A sample C program

```
#include <stdio.h>
#include <stdlib.h>

#include <bsplineval.h>
#include <spline_coef.h>

/*
   This program illustrates the sequences of calls to make to evaluate a
   spline.
*/

int use_selected_Bspline_evaluator( BsplineInput* bspline_input ) {

  double tt = 3.14, zz = 0.0;
  BsplineResult result;

  /* Algorithm B then Algorithm A will be used in the call to Algorithm C */
  bspline_input->method1 = 'b';
  bspline_input->method2 = 'a';

  if ( Bspline_Success !=
       BsplineAlgo_c_derivs( tt, zz, bspline_input, &result ) ) {
    fprintf( stderr,
     "BsplineAlgo_c_derivs returned with an error  t=%.14f, "
     "z=%.14f\n", tt, zz );
    return EXIT_FAILURE;
  }

  return EXIT_SUCCESS;

}

/* This is the recommended used of the call to the library */
int use_optimized_Bspline( BsplineInput* bspline_input ) {

  double tt = 3.14, zz = 0.0;
  BsplineResult result;

  if ( Bspline_Success !=
       Bspline_eval_derivs( tt, zz, bspline_input, &result ) ) {
    fprintf( stderr,
     "Bspline_eval_derivs returned with an error  t=%.14f, "
```

```
      "z=%.14f\n", tt, zz );
    return EXIT_FAILURE;
  }

  return EXIT_SUCCESS;

}

int main( int argc, char** argv ) {

  /* Common c; */
  BsplineInput* bspline_input = NULL;

  /*
   * The user is responsible for supplying the information necessary to
   * evaluate the B-spline: the knots in theta and z, the order of knots etc..
   * This structure must be filled in by the user before making any more calls
   * to the library.
   */
  spline_coef scoef;

  /*
   * It is the user's responsibilty to read in the Bspline coefficients.
   * read_spline_data_( c->input, &input_unit, &c->rms_amplitude );
   */

  /* Allocate the memory used by the library */
  bspline_input = (BsplineInput*) Bspline_init( &scoef );
  if ( NULL == bspline_input ) {
    fprintf( stderr,
     "process( ) : Unable to allocate memory for eval_spline\n" );
    return EXIT_FAILURE;
  }

  /* This is the recommended used of the call to the library */
  if ( EXIT_FAILURE == use_optimized_Bspline( bspline_input ) ) {
    Bspline_free( bspline_input );
    return EXIT_FAILURE;
  }

  /* If the user insists on using one of the slow routines */
  if ( EXIT_FAILURE == use_selected_Bspline_evaluator( bspline_input ) ) {
    Bspline_free( bspline_input );
    return EXIT_FAILURE;
  }

  /* Must free memory which was allocated in the call to Bspline_init */
```

```
  Bspline_free( bspline_input );

  return EXIT_SUCCESS;

}
```

4 Library Routines

4.1 BsplineAlgo_a

BsplineAlgo_a – The evaluation of s(x) from its normalized B-spline representation using Scheme A (repeated convex combinations).

Synopsis

```
#include <BsplineEval/bsplineval.h>
#include <BsplineEval/spline_coef.h>

BsplineErr BsplineAlgo_a(
  double x,
  size_t j_index,
  size_t order,
  double *c,
  double *d,
  double *knot,
  double *result
);
```

Parameters

`double x` The point to evaluate the B-spline at

`size_t j_index`
knot[j_index-1] <= x < knot[j_index], j

`size_t order`
the order of the B-spline, n

`double *c` the B-spline coefficients, subscripted c

`double *d` work space of dim : number of knots + order

`double *knot`
the B-spline knots, subscripted x

`double *result`
s(x)

Description

The Numerical Evaluation of a Spline from its B-Spline Representation by M. G. Cox. J. Inst. Maths. Applics (1978) 21, 135-143.

Returns

The enum BsplineErr.

Possible values for a `BsplineErr` are as follows:

`Bspline_Success`
: Success

`Bspline_OutOfRange`
: One of the arguments is out of range

`Bspline_OutOfRangeT`
: The theta value is not within the limits of the t_knots

`Bspline_OutOfRangeZ`
: The z value is outside the limits of the z_knots

`Bspline_WrongMethod1`
: The method request must be one of : a, A, b or B

`Bspline_WrongMethod2`
: The method request must be one of : a, A, b or B

4.2 BsplineAlgo_b

BsplineAlgo_b: The evaluation of s(x) from its normalized B-spline representation using Scheme B (computation of the basis).

Synopsis

```
#include <BsplineEval/bsplineval.h>
#include <BsplineEval/spline_coef.h>

BsplineErr BsplineAlgo_b(
  double x,
  size_t j_index,
  size_t order,
  double *c,
  double *v,
  double *knot,
  double *result
);
```

Parameters

`double x` The point to evaluate the B-spline at

`size_t j_index`
: knot[j_index-1] <= x < knot[j_index], j

`size_t order`
: the order of the B-spline, n

`double *c`
: the B-spline coefficients, subscripted c

`double *v`
: work space of dimension : number order + 1

`double *knot`
: the B-spline knots, subscripted x

`double *result`
: s(x)

Description

The Numerical Evaluation of a Spline from its B-Spline Representation by M. G. Cox. J. Inst. Maths. Applics (1978) 21, 135-143.

Returns

The enum BsplineErr.

Possible values for a `BsplineErr` are as follows:

`Bspline_Success`
: Success

`Bspline_OutOfRange`
: One of the arguments is out of range

`Bspline_OutOfRangeT`
: The theta value is not within the limits of the t_knots

`Bspline_OutOfRangeZ`
: The z value is outside the limits of the z_knots

`Bspline_WrongMethod1`
: The method request must be one of : a, A, b or B

`Bspline_WrongMethod2`
: The method request must be one of : a, A, b or B

4.3 BsplineAlgo_c

BsplineAlgo_c: The evaluation of s(t,z) from its normalized B-spline representation. This function allows the user to select the underlying algorithm used, 'a' or 'b'.

Synopsis

```
#include <BsplineEval/bsplineval.h>
#include <BsplineEval/spline_coef.h>
```

```
BsplineErr BsplineAlgo_c(
  double theta,
  double z,
  BsplineInput *bspline_input,
  double *result
);
```

Parameters

`double theta`
: The value of theta to evaluate the B-spline

`double z`
: The value of z to evaluate the B-spline

`BsplineInput *bspline_input`
: All neccessary B-spline input

`double *result`
: s(theta, z)

Description

The Numerical Evaluation of a Spline from its B-Spline Representation by M. G. Cox. J. Inst. Maths. Applics (1978) 21, 135-143.

Returns

The enum BsplineErr. SEE ALSO Bspline_eval

Possible values for a `BsplineErr` are as follows:

`Bspline_Success`
: Success

`Bspline_OutOfRange`
: One of the arguments is out of range

`Bspline_OutOfRangeT`
: The theta value is not within the limits of the t_knots

`Bspline_OutOfRangeZ`
: The z value is outside the limits of the z_knots

`Bspline_WrongMethod1`
: The method request must be one of : a, A, b or B

`Bspline_WrongMethod2`
: The method request must be one of : a, A, b or B

4.4 BsplineAlgo_c_derivs

BsplineAlgo_c_derivs – The evaluation of s(t,z) and the derivative with respect to t and z from its normalized B-spline representation. This function allows the user to select the underlying algorithm used, 'a' or 'b'.

Synopsis

```
#include <BsplineEval/bsplineval.h>
#include <BsplineEval/spline_coef.h>

BsplineErr BsplineAlgo_c_derivs(
  double theta,
  double z,
  BsplineInput *bspline_input,
  BsplineResult *bspline_result
);
```

Parameters

`double theta`
: The theta to evaluate the B-spline

`double z`
: The value of z to evaluate the B-spline

`BsplineInput *bspline_input`
: All B-spline input

`BsplineResult *bspline_result`
: All B-spline output

Description

The Numerical Evaluation of a Spline from its B-Spline Representation by M. G. Cox. J. Inst. Maths. Applics (1978) 21, 135-143.

Returns

The enum BsplineErr.

Possible values for a `BsplineErr` are as follows:

`Bspline_Success`
: Success

`Bspline_OutOfRange`
: One of the arguments is out of range

`Bspline_OutOfRangeT`
: The theta value is not within the limits of the t_knots

`Bspline_OutOfRangeZ`
: The z value is outside the limits of the z_knots

`Bspline_WrongMethod1`
: The method request must be one of : a, A, b or B

`Bspline_WrongMethod2`
: The method request must be one of : a, A, b or B

4.5 Bspline_determine_index

1] <= key < base[index]. If key == base[num - 1] then index is set to num - 1.

Synopsis

```
#include <BsplineEval/bsplineval.h>
#include <BsplineEval/spline_coef.h>

BsplineErr Bspline_determine_index(
  double *key,
  const void *base,
  size_t num,
  double minval,
  double maxval,
  size_t *index
);
```

Parameters

`double *key`
: The key to search for

`const void *base`
: Points to the element at the base of array

`size_t num`
: Number of elements in the array

`double minval`
: The left most knot boundary

`double maxval`
: The right most knot boundary

`size_t *index`
: base[index - 1] <= key < base[index]

Description

Do a Binary search to find the location of the key within the knots. If this routine is usually called iteratively between the ideal surface and the deformation map then it can improve a bit by making use of the last position found.

Returns

The enum BsplineErr.

Possible values for a `BsplineErr` are as follows:

`Bspline_Success`
: Success

`Bspline_OutOfRange`
: One of the arguments is out of range

`Bspline_OutOfRangeT`
: The theta value is not within the limits of the t_knots

`Bspline_OutOfRangeZ`
: The z value is outside the limits of the z_knots

`Bspline_WrongMethod1`
: The method request must be one of : a, A, b or B

`Bspline_WrongMethod2`
: The method request must be one of : a, A, b or B

4.6 Bspline_err_msg

Bspline_err_msg – print the error message to the file pointer.

Synopsis

```
#include <BsplineEval/bsplineval.h>
#include <BsplineEval/spline_coef.h>

void Bspline_err_msg(
  BsplineErr err,
  FILE *fp
);
```

Parameters

`BsplineErr err`
: Possible values for a `BsplineErr` are as follows:

  `Bspline_Success`
  : Success

  `Bspline_OutOfRange`
  : One of the arguments is out of range

  `Bspline_OutOfRangeT`
  : The theta value is not within the limits of the t_knots

`Bspline_OutOfRangeZ`
: The z value is outside the limits of the z_knots

`Bspline_WrongMethod1`
: The method request must be one of : a, A, b or B

`Bspline_WrongMethod2`
: The method request must be one of : a, A, b or B

`FILE *fp` Not Documented.

Description

Bspline_err_msg – print the error message to the file pointer.

Returns

void.

4.7 Bspline_eval

Bspline_eval: The evaluation of s(t,z) from its normalized B-spline representation. This function is similar to the function BsplineAlgo_c with the exception that the underlying algorithms 'b' and 'b' are chosen for the user.

Synopsis

```
#include <BsplineEval/bsplineval.h>
#include <BsplineEval/spline_coef.h>

BsplineErr Bspline_eval(
  double theta,
  double z,
  BsplineInput *bspline_input,
  double *result
);
```

Parameters

`double theta`
: The value of theta to evaluate the B-spline

`double z` The value of z to evaluate the B-spline

`BsplineInput *bspline_input`
: All neccessary B-spline input

`double *result`
: s(theta, z)

Description

The Numerical Evaluation of a Spline from its B-Spline Representation by M. G. Cox. J. Inst. Maths. Applics (1978) 21, 135-143.

Returns

The enum BsplineErr.

Possible values for a `BsplineErr` are as follows:

`Bspline_Success`
: Success

`Bspline_OutOfRange`
: One of the arguments is out of range

`Bspline_OutOfRangeT`
: The theta value is not within the limits of the t_knots

`Bspline_OutOfRangeZ`
: The z value is outside the limits of the z_knots

`Bspline_WrongMethod1`
: The method request must be one of : a, A, b or B

`Bspline_WrongMethod2`
: The method request must be one of : a, A, b or B

4.8 Bspline_eval_derivs

Bspline_eval_derivs – The evaluation of s(t,z) and the derivative with respect to t and z from its normalized B-spline representation. This function is an optimized version of the function BsplineAlgo_c_derivs.

Synopsis

```
#include <BsplineEval/bsplineval.h>
#include <BsplineEval/spline_coef.h>

BsplineErr Bspline_eval_derivs(
  double theta,
  double z,
  BsplineInput *bspline_input,
  BsplineResult *bspline_result
);
```

Parameters

`double theta`
: The theta to evaluate the B-spline

`double z` The value of z to evaluate the B-spline

`BsplineInput *bspline_input`
: All B-spline input

`BsplineResult *bspline_result`
: All B-spline output

Description

An optimized implementation of the function BsplineAlgo_c. This function can be further optimized by folding in the last of the seperate call to BsplineAlgo_b, but this requires the rewriting of the algo_b routine. The Numerical Evaluation of a Spline from its B-Spline Representation by M. G. Cox. J. Inst. Maths. Applics (1978) 21, 135-143.

Returns

The enum BsplineErr.

Possible values for a `BsplineErr` are as follows:

`Bspline_Success`
: Success

`Bspline_OutOfRange`
: One of the arguments is out of range

`Bspline_OutOfRangeT`
: The theta value is not within the limits of the t_knots

`Bspline_OutOfRangeZ`
: The z value is outside the limits of the z_knots

`Bspline_WrongMethod1`
: The method request must be one of : a, A, b or B

`Bspline_WrongMethod2`
: The method request must be one of : a, A, b or B

4.9 Bspline_free

Bspline_free – free memory allocated by Bspline_init.

Synopsis

```
#include <BsplineEval/bsplineval.h>
#include <BsplineEval/spline_coef.h>

void Bspline_free(BsplineInput *input);
```

Parameters

`BsplineInput *input`
: Not Documented.

Description

Bspline_free – free memory allocated by Bspline_init.

Returns

void.

4.10 Bspline_init

Allocates the memory used for Bspline library.

Synopsis

```
#include <BsplineEval/bsplineval.h>
#include <BsplineEval/spline_coef.h>

BsplineInput *Bspline_init(spline_coef *scoef);
```

Parameters

`spline_coef *scoef`
: Not Documented.

Description

Allocates the memory used for Bspline library.

Returns

BsplineInput* if successful else NULL is returned.

5 A Timing Comparison of BsplineEval library vs DASL

```
The file ./test/p6.spl with the following characteristics:

# spline deformation amplitudes
# z_order         4
# theta_order     4
# qz              421
# num_z_knots     417
# num_theta_knots 144
# zmin zmax       -1.000000e+00 1.000000e+00
# tmin tmax       0.000000e+00 6.283185e+00
# rms amplitude   129188.64688685712463
#
```

will be used to derive the following timing results for evaluating the spline without evaluating the derivatives and then with derivatives. The actual time one may get is dependent on many factors such as the number of theta and z knots etc...

```
           |        6     |         6    |         6
           |      10      |   4 * 10     |   9 * 10
-----------|--------------|--------------|--------------
   |   |              |
ab    | 35.7      |   1:58.9     |   4:17.5
ba    | 33.6      |   1:50.7     |   3:59.0
aa    | 36.2      |   2:01.4     |   4:23.2
bb    | 32.9      |   1:48.2     |   3:53.3
optimized  |    33.2      |   1:48.9     |   3:55.5
dasl       |    59.8      |   3:35.5     |   7:54.3
```

Note that the time for the optimized version (without evaluating the derivatives) is slightly more then the time for the 'bb' version. The reason for this is the routine `Bspline_eval` was provided as a symmetry to the function `Bspline_eval_derivs`. The routine `Bspline_eval` sets to flag, method1='b'; and method2='b'; before calling routine 'bb' so a slightly slower time is expected.

```
           |        6     |         6    |         6
           |      10      |   4 * 10     |   9 * 10
-----------|--------------|--------------|--------------
   |   |              |
ab    | 1:09.8    |   4:15.6     |   9:24.8
ba    | 1:04.5    |   3:54.5     |   8:37.9
aa    | 1:11.4    |   4:21.9     |   9:39.4
bb    | 1:02.8    |   3:47.7     |   8:22.7
```

optimized	45.5	2:38.1	5:45.6
dasl	2:38.2	10:09.0	22:40.0

The script ./test/timeit.ksh was used to generate the entries to this table.

The time gain in the dasl routine versus the optimized Algorithm C is dependent on several factors. The order of the spline and then number of knots in theta and z.

6 Testing the BsplineEval library vs DASL

Testing was performed by comparing the results given by the dasl and the optimized algorithm C. The paragraphs to follow show the results of two different input files, one with 'large' rms deformation (p6.spl) and one with 'small' rms deformation (p6_low_31_72.SPL). The results agree to within roundoff errors, this was done by running the following command:

```
The file ./test/p6.spl with the following characteristics:

# spline deformation amplitudes
# z_order         4
# theta_order     4
# qz              421
# num_z_knots     417
# num_theta_knots 144
# zmin zmax       -1.000000e+00 1.000000e+00
# tmin tmax       0.000000e+00 6.283185e+00
# rms amplitude   129188.64688685712463
#

yield the following results:

futile-90: objs/sun4u-SunOS-5/evalbspline num=100 test=5 \
| column -v -a deltas N - | column -v -a deltadt N \
| column -v -a deltadz N \
        | compute deltas = \( dasl - algoc \) \/ dasl \
        | compute deltadz = \( ddasldz - dalgocdz \) \/ ddasldz \
        | compute deltadt = \( ddasldt - dalgocdt \) \/ ddasldt \
        | rdbstats deltas deltadz deltadt | tbl2lst

     deltas_n | N
   deltas_sum | N
   deltas_ave | N
   deltas_dev | N
   deltas_min | N
   deltas_max | N
    deltadz_n | N
  deltadz_sum | N
  deltadz_ave | N
  deltadz_dev | N
  deltadz_min | N
  deltadz_max | N
    deltadt_n | N
  deltadt_sum | N
  deltadt_ave | N
  deltadt_dev | N
```

```
  deltadt_min | N
  deltadt_max | N

     deltas_n | 10000
   deltas_sum | -1.29628283548978e-13
   deltas_ave | -1.29628283548978e-17
   deltas_dev | 1.10245039134638e-16
   deltas_min | -5.83761910253855e-16
   deltas_max | 5.3919301269975e-16
    deltadz_n | 10000
  deltadz_sum | -9.28799965664604e-14
  deltadz_ave | -9.28799965664602e-18
  deltadz_dev | 1.33699222283282e-16
  deltadz_min | -3.90195209692813e-15
  deltadz_max | 3.74917737998133e-15
    deltadt_n | 10000
  deltadt_sum | -1.46179231627943e-13
  deltadt_ave | -1.46179231627943e-17
  deltadt_dev | 1.07162272062163e-16
  deltadt_min | -6.82814047720486e-16
  deltadt_max | 7.42207044201643e-16
```

The file ./test/p6_low_31_72.SPL with the following characteristics:

```
# spline deformation amplitudes
# z_order         4
# theta_order     4
# qz              33
# num_z_knots     29
# num_theta_knots 71
# zmin zmax       -1.000000e+00 1.000000e+00
# tmin tmax       0.000000e+00 6.283185e+00
# rms amplitude   0.00101284258284
```

yield the following results:

```
futile-93: objs/sun4u-SunOS-5/evalbspline num=100 test=5 \
input=p6_low_31_72.SPL \
| column -v -a deltas N - | column -v -a deltadt N \
| column -v -a deltadz N \
        | compute deltas = \( dasl - algoc \) \/ dasl \
        | compute deltadz = \( ddasldz - dalgocdz \) \/ ddasldz \
        | compute deltadt = \( ddasldt - dalgocdt \) \/ ddasldt \
        | rdbstats deltas deltadz deltadt | tbl2lst
```

```
   deltas_n | N
 deltas_sum | N
 deltas_ave | N
 deltas_dev | N
 deltas_min | N
 deltas_max | N
  deltadz_n | N
deltadz_sum | N
deltadz_ave | N
deltadz_dev | N
deltadz_min | N
deltadz_max | N
  deltadt_n | N
deltadt_sum | N
deltadt_ave | N
deltadt_dev | N
deltadt_min | N
deltadt_max | N

   deltas_n | 10000
 deltas_sum | -0
 deltas_ave | 0
 deltas_dev | 0
 deltas_min | -0
 deltas_max | -0
  deltadz_n | 10000
deltadz_sum | 0
deltadz_ave | 0
deltadz_dev | 0
deltadz_min | 0
deltadz_max | 0
  deltadt_n | 10000
deltadt_sum | 0
deltadt_ave | 0
deltadt_dev | 0
deltadt_min | -0
deltadt_max | -0
```

7 Known Problems and Improvements to be made

In evaluating s(theta, z) from its normalized representation, the library currently assumes theta to be periodic. The library can 'easily' be made to be more general, ie the library can should allow the user to specify the which variable is periodic or none at all.

The routine eval_Bspline_derivs is an optimized version of the routine BsplineAlgo_c. This was done by folding one of the calls to calculate the derivative into the call to evaluate s(theta, z). It will require a little work to fold in the other derivative. A 10-15% savings is expected.

The library currently needs three arrays of B-spline coefficients (scoef->c, scoef->cdifz, scoef->cdift), a more optimized data structure would be to pass the library an array of structures. Theoretically, this will minimize caching in results since only n number (where n is the order of the B-spline) of elements are needed to evaluate the spline.