Basic Numerical Analysis Routines

Brent Seidel
Phoenix, AZ

July 11, 2024

Contents

1 Introduction **1**
- 1.1 BBS.Numerical.complex . . . 1
- 1.2 BBS.Numerical.derivative . . . 1
- 1.3 BBS.Numerical.functions . . . 1
- 1.4 BBS.Numerical.integration_real . . . 1
- 1.5 BBS.Numerical.interpolation . . . 2
- 1.6 BBS.Numerical.ode . . . 2
- 1.7 BBS.Numerical.polynomial_complex . . . 2
- 1.8 BBS.Numerical.polynomial_real . . . 2
- 1.9 BBS.Numerical.quaternion . . . 2
- 1.10 BBS.Numerical.random . . . 2
- 1.11 BBS.Numerical.regression . . . 2
- 1.12 BBS.Numerical.roots_complex . . . 2
- 1.13 BBS.Numerical.roots_real . . . 2
- 1.14 BBS.Numerical.statistics . . . 3
- 1.15 BBS.Numerical.vector . . . 3

2 How to Obtain **4**
- 2.1 Dependencies . . . 4
  - 2.1.1 Ada Libraries . . . 4
  - 2.1.2 Other Libraries . . . 4

3 Usage Instructions **5**

4 API Description **6**
- 4.1 BBS.Numerical.complex . . . 6
- 4.2 BBS.Numerical.derivative . . . 6
  - 4.2.1 Two Point Formulas . . . 6
  - 4.2.2 Three Point Formulas . . . 6
  - 4.2.3 Five Point Formulas . . . 7
- 4.3 BBS.Numerical.functions . . . 7
  - 4.3.1 Gamma Function Related . . . 7
  - 4.3.2 Factorial Related . . . 8
  - 4.3.3 Combinatorial Related . . . 8
- 4.4 BBS.Numerical.integration_real . . . 8

4.5 BBS.Numerical.interpolation 9
4.6 BBS.Numerical.ode 9
4.7 BBS.Numerical.polynomial_complex 9
4.8 BBS.Numerical.polynomial_real 9
4.9 BBS.Numerical.quaternion 9
4.10 BBS.Numerical.random 9
4.11 BBS.Numerical.regression 9
4.12 BBS.Numerical.roots_complex 9
4.13 BBS.Numerical.roots_real 9
4.14 BBS.Numerical.statistics 9
4.15 BBS.Numerical.vector 9

Chapter 1

Introduction

Back in the 1980s when I was an undergraduate, I took a numerical analysis course and quite enjoyed it. Then my first job out of college was working on a numerical analysis library for a small startup that went the way of most startups. I was recently inspired to dig out my old textbook and try implementing some of the routines. This collection includes some of those, plus others.

Note that some packages are for complex numbers and some are for real numbers. At some point, they may be combined. Most packages are generic. The packages are:

1.1 BBS.Numerical.complex

This is an object oriented collection of complex number routines. After writing this, I discovered `Ada.Numerics.Generic_Complex_Types`. So this package is deprecated in favor of the Ada package.

1.2 BBS.Numerical.derivative

This is a generic package with a real type parameter. It contains functions to compute the derivative of real valued functions with a single argument.

1.3 BBS.Numerical.functions

This is a generic package with a real type parameter. It contains some functions that are used by other packages.

1.4 BBS.Numerical.integration_real

This is a generic package with a real type parameter. It contains functions to compute integrals of real valued functions with a single argument.

1.5 BBS.Numerical.interpolation

This is a generic package with a real type parameter. It contains functions to interpolate a value between a set of data points. The functions can also extrapolate (if using a value of x outside of the range of the data points, however this is likely to be inaccurate.

1.6 BBS.Numerical.ode

This is a generic package with a real type parameter. It contains functions to solve ordinary differential equations.

1.7 BBS.Numerical.polynomial_complex

This is a generic package with a complex type parameter (from `Ada.Numerics.Generic_Complex_Types`. It contains functions for polynomials.

1.8 BBS.Numerical.polynomial_real

This is a generic package with a real type parameter. It contains functions for polynomials.

1.9 BBS.Numerical.quaternion

This is a generic package with a real type parameter. It contains functions for quaternions.

1.10 BBS.Numerical.random

This is not a generic package. It contains functions for generating pseudo-random numbers.

1.11 BBS.Numerical.regression

This is a generic package with a real type parameter. It contains functions for performing regression analysis of data.

1.12 BBS.Numerical.roots_complex

This is a generic package with a complex type parameter (from `Ada.Numerics.Generic_Complex_Types`. It contains functions for finding zeros of functions.

1.13 BBS.Numerical.roots_real

This is a generic package with a real type parameter. It contains functions for finding zeros of functions.

1.14 BBS.Numerical.statistics

This is a generic package with a real type parameter. It contains statistics related functions.

1.15 BBS.Numerical.vector

This is a generic package with a real type parameter. It contains functions for vectors.

Chapter 2

How to Obtain

This collections is currently available on GitHub at `https://github.com/BrentSeidel/Numerical`.

2.1 Dependencies

2.1.1 Ada Libraries

The following Ada libraries are used:

- `Ada.Numerics`
- `Ada.Numerics.Generic_Complex_Types`
- `Ada.Numerics.Generic_Complex_Elementary_Functions`
- `Ada.Numerics.Generic_Elementary_Functions`
- `Ada.Text_IO` (used for debugging purposes)

2.1.2 Other Libraries

This library depends on the root package `BBS` available at `https://github.com/BrentSeidel/BBS-Ada`. Internal packages used within this library are:

- `BBS.Numerical.functions`
- `BBS.Numerical.Integration_real`

Chapter 3

Usage Instructions

This is a library of routines intended to be used by some program. To use these in your program, edit your *.gpr file to include a line to `with` the path to `BBS_Numerical.gpr`. Then in your Ada code `with` in the package(s) you need and use the routines.

Chapter 4

API Description

4.1 BBS.Numerical.complex

This package is deprecated in favor of the Ada package `Ada.Numerics.Generic_Complex_Types`.

4.2 BBS.Numerical.derivative

This is a generic package with a real type parameter, `F`.

It defines a function type `test_func` as `access function (x : f'Base) return f'Base`.

WARNING: The calculations here may involve adding small numbers to large numbers and taking the difference of two nearly equal numbers. The world of computers is not the world of mathematics where numbers have infinite precision. If you aren't careful, you can get into a situation where $(x + h) = x$, or $f(x) = f(x \pm h)$. For example assume that the float type has 6 digits. Then, if x is $1,000,000$ and h is 1, adding x and h is a wasted operation.

The functions are:

4.2.1 Two Point Formulas

Two point formula. Use $h > 0$ for forward-difference and $h < 0$ for backward-difference. Derivative calculated at point x. While this is the basis for defining the derivative in calculus, it generally shouldn't be used.

```
pt2(f1 : test_func; x, h : f'Base) return f'Base
```

- `f1` - the function that you with to find the derivative of
- `x` - the point at which to calculate the derivative
- `h` - the step size

4.2.2 Three Point Formulas

Compute the derivative at x using values at x, $x + h$, and $x + 2h$.

```
pt3a(f1 : test_func; x, h : f'Base) return f'Base
```

- `f1` - the function that you with to find the derivative of
- `x` - the point at which to calculate the derivative
- `h` - the step size

Compute the derivative at x using values at $x - h$ and $x + h$.
`pt3b(f1 : test_func; x, h : f'Base) return f'Base`

- `f1` - the function that you with to find the derivative of
- `x` - the point at which to calculate the derivative
- `h` - the step size

4.2.3 Five Point Formulas

Compute the derivative at x using values at $x - 2h$, $x - h$, $x + h$, and $x + 2h$.
`pt5a(f1 : test_func; x, h : f'Base) return f'Base`

- `f1` - the function that you with to find the derivative of
- `x` - the point at which to calculate the derivative
- `h` - the step size

Compute the derivative at x using values at x, $x + h$, $x + 2h$, $x + 3h$, and $x + 4h$.
`pt5b(f1 : test_func; x, h : f'Base) return f'Base`

- `f1` - the function that you with to find the derivative of
- `x` - the point at which to calculate the derivative
- `h` - the step size

4.3 BBS.Numerical.functions

This is a generic package with a real type parameter, `F`. Its primary purpose is to contain functions that are used by other packages in this library. They may also be useful to the end user.

4.3.1 Gamma Function Related

Some of the probability functions require computing $\Gamma(\frac{n}{2})$. Since $\Gamma(\frac{1}{2}) = \sqrt{pi}$, $\Gamma(1) = 1$, and $\Gamma(a + 1) = a * \Gamma(a)$, computing $\Gamma(\frac{n}{2})$, where n is a positive integer is simpler than computing the full Γ function.

Computes Γ of $2n$. Note that this will overflow for `Float` for $n > 70$.
`gamma2n(n : Positive) return f'Base`

- n - A positive integer representing twice the value for computing Γ.
- Returns the actual value.

To compute Γ of $2n$ where $n > 70$, logarithms can be used. This returns the $log_e(\Gamma(n))$.
`lngamma2n(n : Positive) return f'Base`

- n - A positive integer representing twice the value for computing Γ.
- Returns the natural log of the value.

4.3.2 Factorial Related

The factorial functions are similar to the Γ functions. Since the parameter is not divided by two in this case, overflow will occur for $n > 35$. Thus, as for Γ, there are two versions. The first returns the actual value, the second returns the natural log of the value.
`factorial(n : Natural) return f'Base`

- n - A natural integer representing.
- Returns the actual factorial value.

`lnfact(n : Natural) return f'Base`

- n - A natural integer representing.
- Returns the natural log of the factorial value.

4.3.3 Combinatorial Related

The one function computes the binomial coefficients (aka N choose K). By definition, this computes:

$$\binom{n}{k} = \frac{n!}{k!(b-k)!}$$

With some algebra, this becomes:

$$\binom{n}{k} = \prod_{i=1}^{k} \frac{n+1-i}{i}$$

which is easier to compute. The function is:
`nChoosek(n, k : Natural) return f'Base`

- n - The number of items.
- k - The number of items to choose.
- There is a restriction that $n \geq k$

4.4 BBS.Numerical.integration_real

This package contains routines to perform numerical integration of a function with one parameter. Thus it computes an approximation to:

$$y = \int_a^b f(x)dx$$

4.5 BBS.Numerical.interpolation

4.6 BBS.Numerical.ode

4.7 BBS.Numerical.polynomial_complex

4.8 BBS.Numerical.polynomial_real

4.9 BBS.Numerical.quaternion

4.10 BBS.Numerical.random

4.11 BBS.Numerical.regression

4.12 BBS.Numerical.roots_complex

4.13 BBS.Numerical.roots_real

4.14 BBS.Numerical.statistics

4.15 BBS.Numerical.vector