Basic Numerical Analysis Routines

Brent Seidel
Phoenix, AZ

July 11, 2024

Note that this is a draft version and not the final version for publication.

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 2

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 2

1.15 BBS.Numerical.vector 2

2 How to Obtain 3

2.1 Dependencies 3

2.1.1 Ada Libraries 3

2.1.2 Other Libraries 3

3 Build Instructions 4

4 API Description 5

4.1 BBS.Numerical.complex 5

4.2 BBS.Numerical.derivative 5

4.2.1 Two Point Formulas 5

4.2.2 Three Point Formulas 5

4.2.3 Five Point Formulas 6

4.3 BBS.Numerical.functions 7

4.4 BBS.Numerical.integration_real 7

4.5 BBS.Numerical.interpolation 7

4.6 BBS.Numerical.ode 7

4.7 BBS.Numerical.polynomial_complex 7

4.8 BBS.Numerical.polynomial_real 7
4.9 BBS.Numerical.quaternion 7
4.10 BBS.Numerical.random 7
4.11 BBS.Numerical.regression 7
4.12 BBS.Numerical.roots_complex 7
4.13 BBS.Numerical.roots_real 7
4.14 BBS.Numerical.statistics 7
4.15 BBS.Numerical.vector 7

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 package contains some functions that are used by other packages.

1.4 BBS.Numerical.integration_real

1.5 BBS.Numerical.interpolation

1.6 BBS.Numerical.ode

1.7 BBS.Numerical.polynomial_complex

1.8 BBS.Numerical.polynomial_real

1.9 BBS.Numerical.quaternion

1.10 BBS.Numerical.random

1.11 BBS.Numerical.regression

1.12 BBS.Numerical.roots_complex

1.13 BBS.Numerical.roots_real

1.14 BBS.Numerical.statistics

1.15 BBS.Numerical.vector

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

2.1.2 Other Libraries

Chapter 3

Build Instructions

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 $f(x)=f(x+/-h)$. 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

4.4 BBS.Numerical.integration_real

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