Table of Contents

Miscellaneous

- Matrix-Vector multiplication (matvec)
- Dot Product (dotproduct)

Machine Precision

Single-Precision Machine Epsilon Value

Routine Name: smaceps

Author: Bryan Armenta

Language: C. The code can be compiled using the GNU C compiler (GCC).

Description/Purpose: Calculates the machine-precision value of a single-precision floating-point number.

Input: No input.

Output: Returns the machine epsilon value of a single-precision float.

Usage/Example:

```
printf("%E", smaceps());
```

Output:

Implementation/Code: The following is the code for smaceps.c:

```
float smaceps()
{
float eps = 1.0f;
float prevEps;   // Stores the last eps such that 1.0f + eps != 1.0f
while (1.0f + eps != 1.0f)
{
prevEps = eps;
eps /= 2.0f;
}
return prevEps;
}
```

Last Modified: October/2023

Double-Precision Machine Epsilon Value

Routine Name: dmaceps

Author: Bryan Armenta

Language: C. The code can be compiled using the GNU C compiler (GCC).

Description/Purpose: Calculates the machine-precision value of a double-precision floating point number.

Input: No input.

Output: Returns the machine epsilon value of a double-precision floating point number.

Usage/Example:

```
printf("%E", dmaceps());
```

Output:

Implementation/Code: The following is the code for smaceps.c:

```
float dmaceps()
{
double eps = 1.0;
double prevEps;   // Stores the last eps such that 1.0 + eps != 1.0
while (1.0 + eps != 1.0)
{
prevEps = eps;
eps /= 2.0;
}
return prevEps;
}
```

Last Modified: October/2023

Norm Lengths & Distances

L1 Norm Length

Routine Name: l1norm

Author: Bryan Armenta

Language: C. The code can be compiled using the GNU C compiler (GCC).

Description/Purpose: Computes the l1-norm length of a vector.

Input: The vector (v) to compute the l1-norm length of, along with its size n.

Output: The l1-norm length of v.

Usage/Example:

Implementation/Code: The following is the code for ...

```
#include <math.h>

double l1norm(double v[], int n)
{
double sum = 0.0;
for (int i = 0; i < n; i++)
{
sum += fabs(v[i]);    // sum |v_i|
}
return sum;
}
```

Last Modified: October/2023

L2 Norm Length

Routine Name: l2norm

Author: Bryan Armenta

Language: C. The code can be compiled using the GNU C compiler (GCC).

Description/Purpose: Computes the l2-norm (Euclidean) length of a vector.

Input: The vector (v) to calculate the length of, along with its size n.

Output: The Euclidean length of the vector v.

Usage/Example:

Implementation/Code: The following is the code for ...

```
#include <math.h>

double l2norm(double v[], int n)
{
double sum = 0.0;    // norm^2
for (int i = 0; i < n; i++)
{
sum += v[i] * v[i]);
}
return sqrt(sum);
}
```

Last Modified: October/2023

L-Infinity Norm Length

Routine Name: linfnorm

Author: Bryan Armenta

Language: C. The code can be compiled using the GNU C compiler (GCC).

Description/Purpose: Computes the infinity-norm length of a vector.

Input: The vector (v) to calculate the length of, along with its size n.

Output: The infinity-norm length of the vector v.

Usage/Example:

Implementation/Code: The following is the code for ...

```
#include <math.h>

double linfnorm(double v[], int n)
{
	double maxAbs = 0.0;
	for (int i = 0; i < n; i++)
	{
		double abs = fabs(v[i]);
		if (x > norm)
		{
			maxAbs = abs;
		}
	}
	return norm;
}
```

Last Modified: October/2023

L1 Norm Distance

Routine Name: l1distance

Author: Bryan Armenta

Language: C. The code can be compiled using the GNU C compiler (GCC).

Description/Purpose: Calculates the distance between two vectors in the L1 norm.

Input: The vectors v1 and v2 to calculate the distance between, along with their size n.

Output: The l1 norm distance between the vectors.

Usage/Example:

Implementation/Code: The following is the code for ...

```
#include <math.h>

double l1distance(double v1[], double v2[], int n)
{
double sum = 0.0;
for (int i = 0; i < n; i++)
{
sum += fabs(v1[i] - v2[i]);
}
return sum;
}
```

Last Modified: October/2023

L2 Norm Distance

Routine Name: l2distance

Author: Bryan Armenta

Language: C. The code can be compiled using the GNU C compiler (GCC).

Description/Purpose: Calculates the distance between two vectors in the L2 norm.

Input: The vectors v1 and v2 to calculate the distance between, along with their size n.

Output: The l2 norm distance between the vectors.

Usage/Example:

Implementation/Code: The following is the code for ...

```
#include <math.h>

double l2distance(double v1[], double v2[], int n)
{
double sum = 0.0;
for (int i = 0; i < n; i++)
{
double x = v1[i] - v2[i];
sum += x * x;
}
return sqrt(sum);
```

Last Modified: October/2023

L-Infinity Norm Distance

Routine Name: linfdistance

Author: Bryan Armenta

Language: C. The code can be compiled using the GNU C compiler (GCC).

Description/Purpose: Calculates the distance between two vectors in the L-infinity norm.

Input: The vectors v1 and v2 to calculate the distance between, along with their size n.

Output: The l-infinity norm distance between the vectors.

Usage/Example:

Implementation/Code: The following is the code for ...

```
#include <math.h>

double l2distance(double v1[], double v2[], int n)
{
double sum = 0.0;
for (int i = 0; i < n; i++)
{
double x = v1[i] - v2[i];
sum += x * x;
}
return sqrt(sum);
```

Last Modified: October/2023

Derivative Approximations

Forward Difference Quotient

Routine Name: forwarddf

Author: Bryan Armenta

Language: C. The code can be compiled using the GNU C compiler (GCC).

Description/Purpose: Approximates f'(x) utilizing the forward difference quotient with increment h.

Input: The function f, the value of x at which to approximate f'(x), and the increment h.

Output: An approximation of f'(x).

Usage/Example:

Implementation/Code: The following is the code for ...

```
double forwarddf(double (*f)(double), double x, double h)
{
return (f(x + h) - f(x)) / h;
}
```

Last Modified: October/2023

Backward Difference Quotient

Routine Name: backwarddf

Author: Bryan Armenta

Language: C. The code can be compiled using the GNU C compiler (GCC).

Description/Purpose: Approximates f'(x) utilizing the backward difference quotient with increment h.

Input: The function f, the value of x at which to approximate f'(x), and the increment h.

Output: An approximation of f'(x).

Usage/Example:

Implementation/Code: The following is the code for ...

```
double backwardDf(double (*f)(double), double x, double h)
{
return (f(x) - f(x - h)) / h;
}
```

Last Modified: October/2023

Central Difference Quotient

Routine Name: centraldf

Author: Bryan Armenta

Language: C. The code can be compiled using the GNU C compiler (GCC).

Description/Purpose: Approximates f'(x) utilizing the central difference quotient with increment h.

Input: The function f, the value of x at which to approximate f'(x), and the increment h.

Output: An approximation of f'(x).

Usage/Example:

Implementation/Code: The following is the code for ...

```
double centraldf(double (*f)(double), double(x), double h)
{
return ((f(x + h) - f(x - h)) / (2 * h));
}
```

Last Modified: October/2023

Linear Systems of Equations

Gauss-Jordan Elimination

Routine Name: gaussjordan

Author: Bryan Armenta

Language: C. The code can be compiled using the GNU C compiler (GCC).

Description/Purpose: Given a system of equations ax=b, reduces the matrix a into upper-triangular form while modifying the solution vector b (utilizing Gauss-Jordan elimination).

Input: A matrix a and a vector b that are related by a system of equations ax = b. The size n of a and b.

Output: No return value, although the matrix a will be modified into upper-triangular form (without leading zeroes) and the vector b will be modified such that ax=b still holds.

Usage/Example: Can be utilized in conjunction with the `backsubstitution()` routine to solve a system of linear equations. For example:

Implementation/Code: The following is the code for ...

```
void gaussjordan(int n, double a[][n], double b[])
{
    // Make a upper triangular (modifying b accordingly)
    for (int k = 0; k < n - 1; k++)
    {
        for (int i = k + 1; i < n; i++)
        {
            double factor = a[i][k] / a[k][k];
            for (int j = k + 1; j < n; j++)
            {
                a[i][j] -= factor * a[k][j];
            }
            b[i] -= factor * b[k];
        }
    }
}
```

Last Modified: October/2023

LU Factorization

Routine Name: lufactorize

Author: Bryan Armenta

Language: C. The code can be compiled using the GNU C compiler (GCC).

Description/Purpose: LU factorizes a matrix in place. Upper-right and diagonal elements are those of the U matrix, while the lower-left elements are those of the L matrix (which has diagonal values of 1).

Input: The matrix a to factorize, along with its size n.

Output: The matrix a will be modified into an in-place lu-factorized form (refer to Description/Purpose).

Usage/Example: Can be used in conjunction with forward substitution and backward substitution to solve a system of linear equations.

Implementation/Code: The following is the code for ...

```
void lufactorize(int n, double a[][n])
{
    for (int k = 0; k < n - 1; k++)
    {
        for (int i = k + 1; i < n; i++)
        {
            double factor = a[i][k] / a[k][k];
            for (int j = k + 1; j < n; j++)
            {
                a[i][j] -= factor * a[k][j];
            }
            a[i][k] = factor;
        }
    }
}
```

Backward Substitution

Routine Name: backsub

Author: Bryan Armenta

Language: C. The code can be compiled using the GNU C compiler (GCC).

Description/Purpose: Utilizes back-substitution to solve for x in ax=b, where a is upper triangular.

Input: Upper triangular matrix a, vector b, vector x (empty), and the size n of all matrices and vectors mentioned.

Output: Fills the contents of the vector x with the solution to ax = b.

Usage/Example:

Implementation/Code: The following is the code for ...

```
void backsub(int n, double a[][n], double b[], double x[])
{
```

```
    x[n - 1] = b[n - 1] / a[n - 1][n - 1];
    for (int i = n - 2; i >= 0; i--)
    {
        double sum = 0.0;
        for (int j = i + 1; j < n; j++)
        {
            sum += a[i][j] * x[j];
        }
        x[i] = (b[i] - sum) / a[i][i];
    }
}
```

Last Modified: October/2023

Forward Substitution

Routine Name: forwardsub

Author: Bryan Armenta

Language: C. The code can be compiled using the GNU C compiler (GCC).

Description/Purpose: Utilizes forward substitution to solve a system of equations of the form ly = b, where l is a lower triangular matrix with diagonal values of 1.

Input: Lower triangular matrix l (diagonal values of 1), vector b, vector y (empty), and the size n of these inputs.

Output: The input vector y will be filled with the solution to ly = b.

Usage/Example:

Implementation/Code: The following is the code for ...

```
void forwardsub(int n, double l[][n], double b[], double y[])
{
    y[0] = b[0];
    for (int i = 0; i < n; i++)
    {
        double sum = 0.0;
        for (int j = 0; j < i; j++)
        {
            sum += l[i][j] * y[j];
        }
        y[i] = b[i] - sum;
    }
}
```

Last Modified: October/2023

Jacobi Iteration

Gauss-Seidel Method

Statistics

Linear Regression

Routine Name: linreg

Author: Bryan Armenta

Language: C. The code can be compiled using the GNU C compiler (GCC).

Description/Purpose: Calculates the coefficients for the linear regression of a vector of y values against a vector of x values.

Input: The vector x of x values, the vector y of y values, the vector A to store the results of the linear regression, and the size n of the x and y vectors.

Output: The vector A will be filled with the coefficients of the linear regression y = ax + b, where A[0] = a and A[1] = b.

Usage/Example:

Implementation/Code: The following is the code for ...

```
void linreg(double x[], double y[], int n, double A[]) {
	double s1 = 0;   // sum x_i
	double s2 = 0;   // sum x_i^2
	
	for (int i = 0; i < n; i++)
	{
		s1 += x[i];
		s2 += x[i] * x[i];
	}
	
	// y = mx + b
	A[0] = 0;   // m
	A[1] = 1;   // b
	
	// (X^T X)^-1 X^T Y
	double det = n * s2 - s1 * s1;
	for (int i = 0; i < n; i++)
	{
		A[0] += y[i] * (n * x[i] - s1);
		A[1] += y[i] * (s2 - s1 * x[i]);
	}
	A[0] /= det;
	A[1] /= det;
}
```

Root Finding

Fixed Point Iteration

Bisection Method

Newton Method

Secant Method

Hybrid Bisection Secant Method

Eigenvalues

Power Method

Inverse Power Method

Shifted Inverse Power Method

Miscellaneous

Matrix-Vector Multiplication

Dot Product
