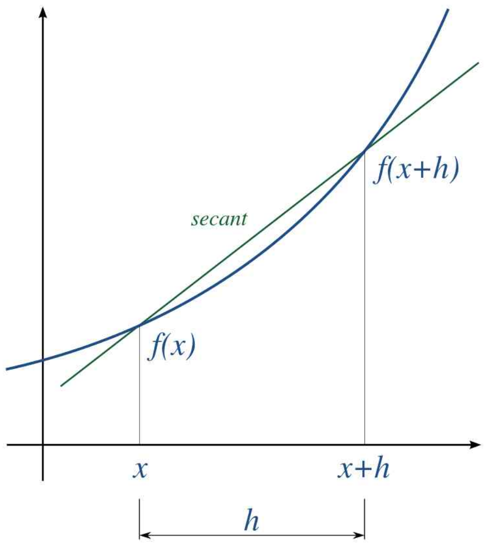
Numerical Differentiation



Newton's difference quotient(== first-order divided difference)

$$\frac{f(x+h)-f(x)}{h}$$

derivative of f at x

$$f'(x)=\lim_{h\to 0}\frac{f(x+h)-f(x)}{h}$$

Symmetric difference quotient

$$\frac{f(x+h)-f(x-h)}{2h}$$

Order of Accuracy

The numerical solution u_h is said to be **nth-order accurate** if the error, $E(h) := ||u-u_h||$ is proportional to the step-size **h** to the **n-th** power

$$E(h) = ||u-u_h|| \leq Ch^n$$

Implementation

```
double NewtonsDifference( FUNCTION f, double x, double dx = 0.0001 )
{
    const double y0 = f( x );
    const double y1 = f( x + dx );
    return ( y1 - y0 ) / dx;
}

double SymmetricDifference( FUNCTION f, double x, double dx = 0.0001 )
{
    const double y0 = f( x - dx );
    const double y1 = f( x + dx );
    return ( y1 - y0 ) / (2.0*dx);
}
```

Drawing

Drawing a Single Variable Function

```
void DrawFunction(HDC hdc, FUNCTION Callback, double beginX, double endX, double xstep, COLORREF color)
{
    double oldX;
    double oldY;
```

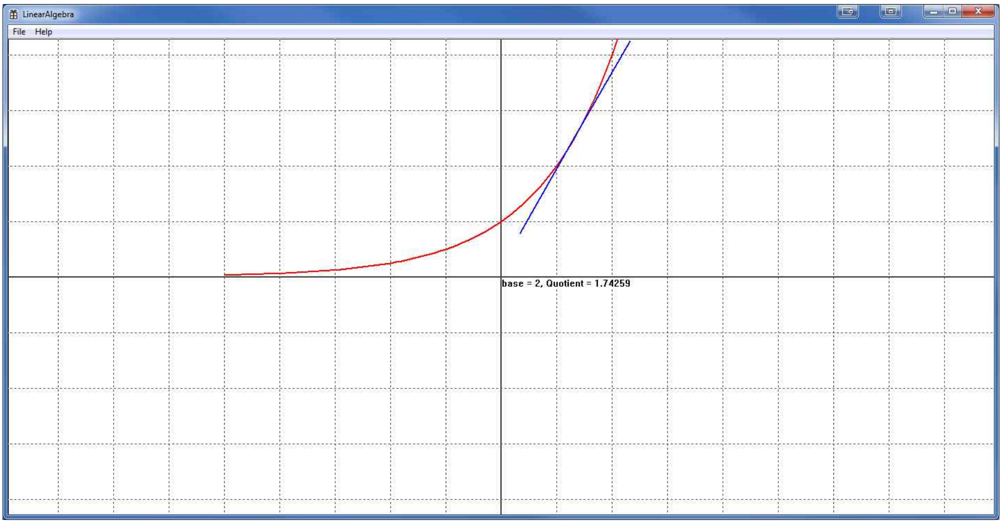
```
    double x = beginX;
    double y = ExpCallback(base_, x);
    oldX = x;
    oldY = y;
    while (x < endX)
    {
        x += xstep;
        y = Callback(x);
        KVectorUtil::DrawLine(hdc, KVector2(oldX, oldY), KVector2(x, y), 2, PS_SOLID,
color);
        oldX = x;
        oldY = y;
    }//while
}
```

Drawing a Tangent Line Segment

```
    const double dx = 0.001;
    double y = Function( x );
    double diff = SymmetricDifference( &Function, x, dx );
    KVector2 v0 = KVector2( x, y );
    KVector2 vdir = KVector2( dx, diff*dx );
    vdir.Normalize( );

    KVectorUtil::DrawLine( hdc, v0, v0 + vdir * 2.0, 2, PS_SOLID, RGB( 0, 0, 255 )
);
    KVectorUtil::DrawLine( hdc, v0, v0 + vdir * -2.0, 2, PS_SOLID, RGB( 0, 0, 255 )
);
```

Result

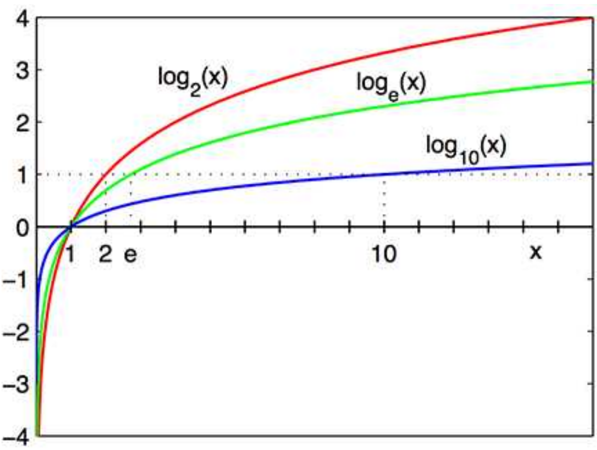


[Fig] Tangent line segment at x

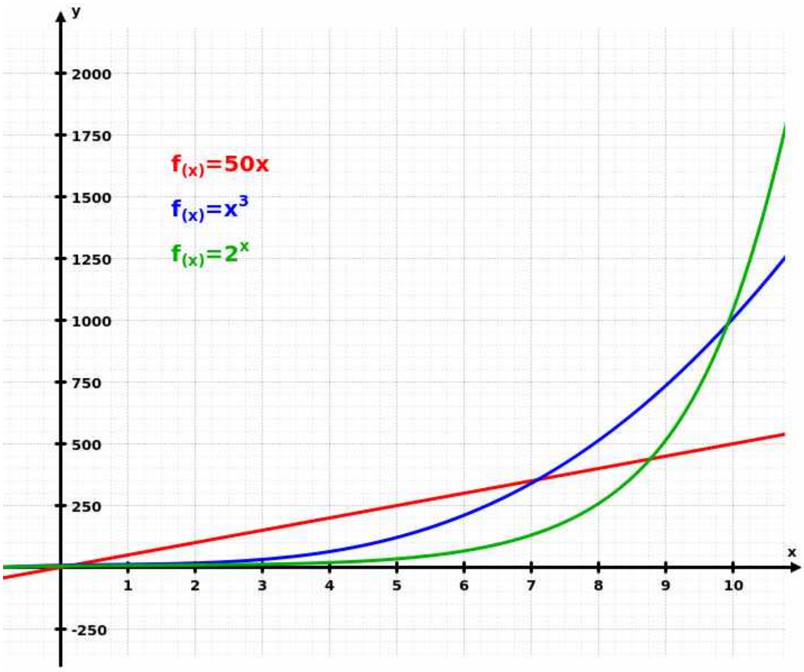
Euler Constant

Logarithm

$$\log_b(x) = y, \;\; b^y = x$$



Why Exponential



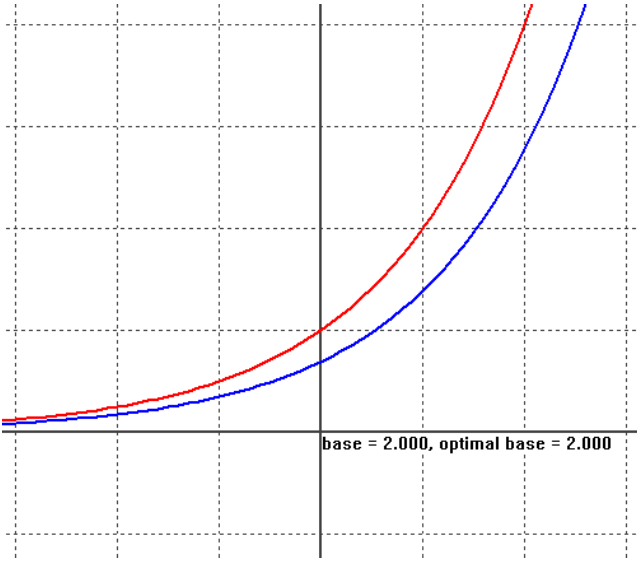
Exponential growth (green) describes many physical phenomena.

Differentiation of exponential function

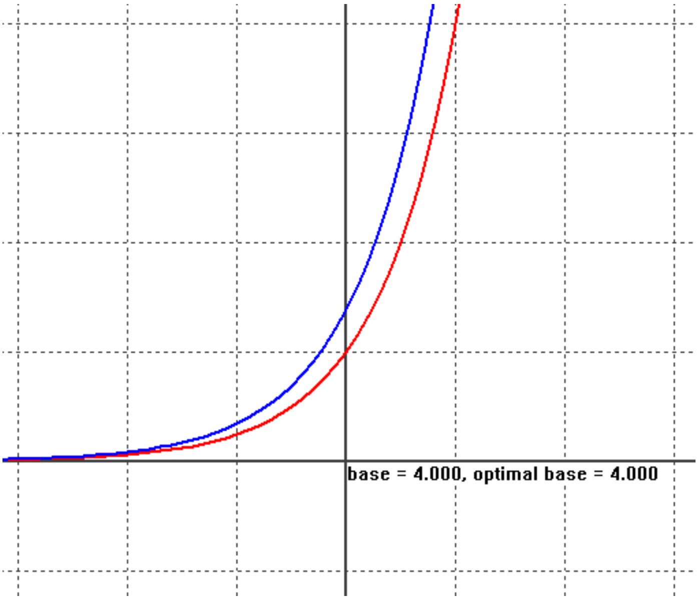
$$\frac{d}{dx}a^x = \lim_{h \to 0} \frac{a^{x+h} - a^x}{h}$$

$$= \lim_{h \to 0} \frac{a^x a^h - a^x}{h}$$

$$= a^x (\lim_{h \to 0} \frac{a^h - 1}{h})$$



[Fig] $2^x, \dfrac{d}{dx}2^x$



[Fig] $4^x, \dfrac{d}{dx}4^x$

$$2 < MysteriousIrrationalConstant\ \ e < 4$$

Numerical solution

Implementation

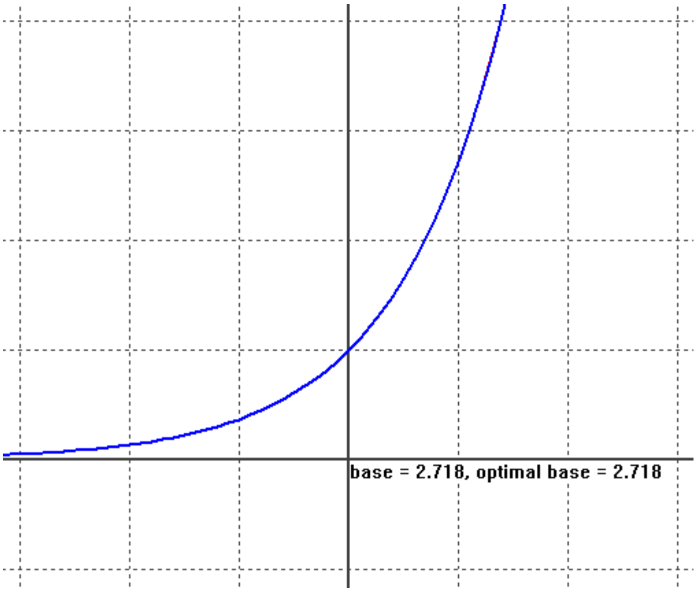
$$s = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \overline{x})^2}{N-1}}$$

```
double GetStdDeviation( double base_, double beginX, double endX, double xstep
)
{
    std::vector<double>     vecDiff;

    double x = beginX;
    double ydiff;
    double N = 0;
    while(x < endX)
    {
        ydiff = ExpFunction( base_, x ) - SymmetricDifference( &ExpFunction,
base_, x );
        x += xstep;
        N += 1;
        vecDiff.push_back( ydiff );
    }//while

    double sum = 0;
    for(const double diff : vecDiff)
    {
        sum += ( diff * diff );
    }

    return sqrt( sum / ( N - 1 ) );
}
```



$e \approx 2.718$

Why e

$$\frac{d}{dx}e^x = e^x$$

$a^x = e^{cx}$

$2^x = e^{x\ln(2)}$

$3^x = e^{x\ln(3)}$

$\ldots$

$$\frac{d}{dx}e^{cx} = ce^{cx}$$

$$\frac{d}{dx}a^x = a^x \ln a$$

Complex Number
Quaternion
Fourier Transform

$$\frac{d}{dx} log_e x = \frac{1}{x}$$

Meaning of e

Example

$$(1+1) = 2$$

$$(1+\frac{1}{2})(1+\frac{1}{2}) = 2.25$$

$$(1+\frac{1}{3})(1+\frac{1}{3})(1+\frac{1}{3}) = 2.370$$

$$(1+\frac{1}{4})(1+\frac{1}{4})(1+\frac{1}{4})(1+\frac{1}{4}) = 2.441$$

...

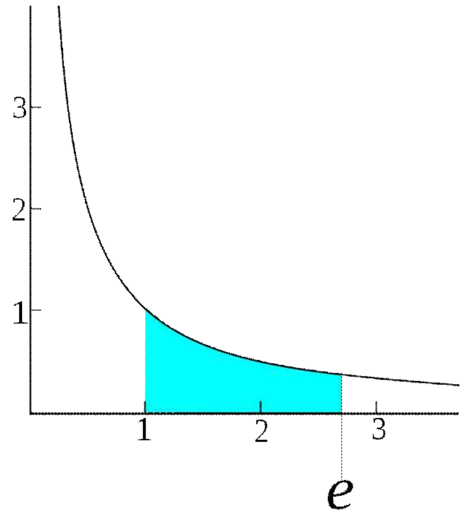
Definition

$$e = \lim_{n \to \infty} (1+\frac{1}{n})^n$$

$$e = \lim_{t \to \infty} (1+t)^{\frac{1}{t}}$$

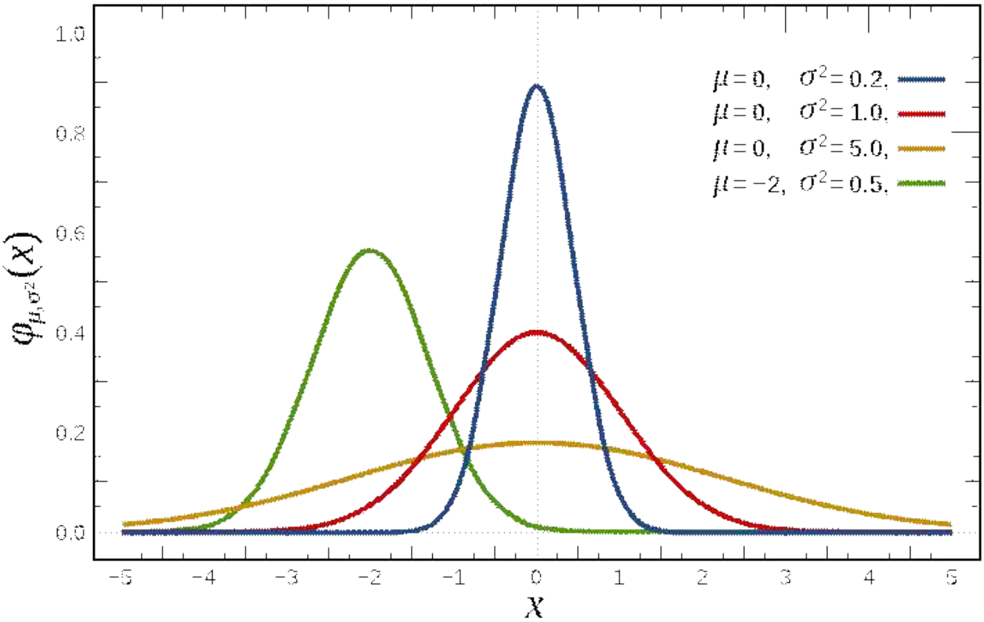
$$e = \sum_{n=0}^{\infty} \frac{1}{n!} = \frac{1}{1} + \frac{1}{1} + \frac{1}{1 \cdot 2} + \frac{1}{1 \cdot 2 \cdot 3} + \dots$$

$$\int_{1}^{e} \frac{1}{t} dt = 1$$



Gaussian function

$$f(x) = ae^{-\frac{(x-b)^2}{2c^2}}$$



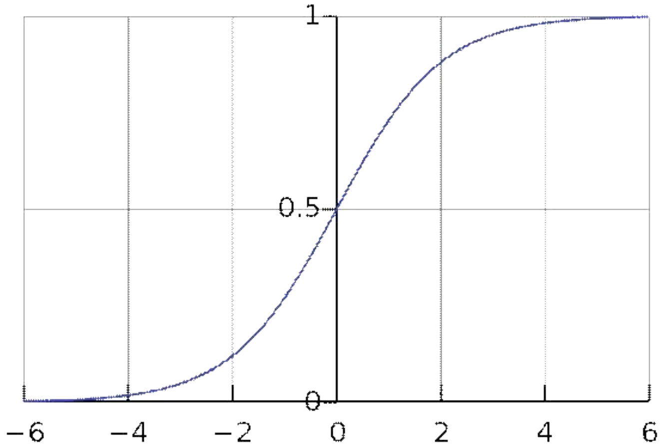
Standard normal distribution

$$S(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$$

Logistic Function

Sigmoid curve

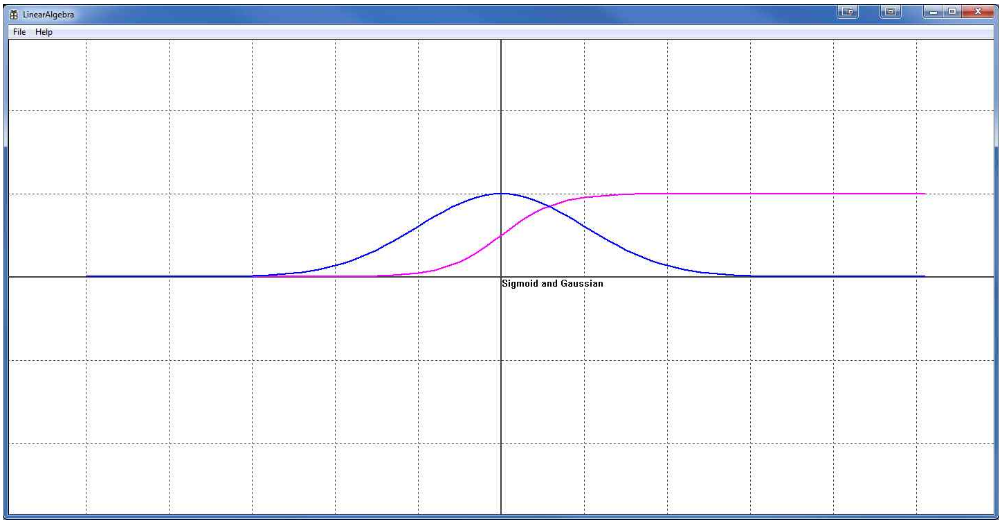
$$f(x) = \frac{L}{1 + e^{-k(x - x_0)}}$$



Implementation

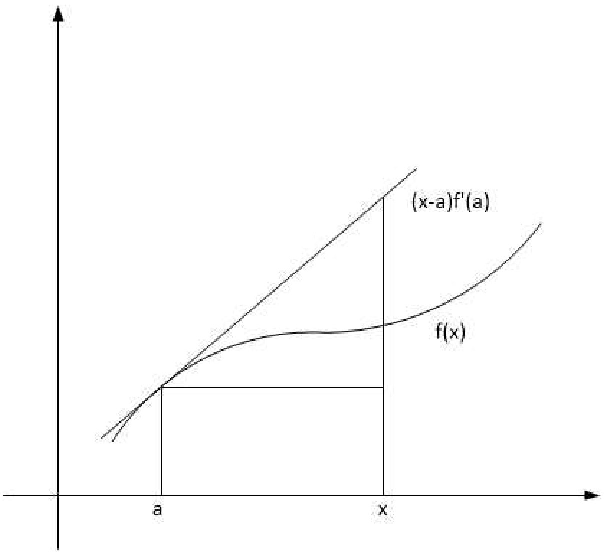
```
double Logistic( double x )
{
    // Logistic Function
    const double L = 1.0;
    const double k = 3.0;
    const double x0 = 0.0;
    return L / (1 + std::exp(-k*(x - x0)));
}

double Gaussian( double x )
{
    // Gaussian Function
    const double a = 1.0;
    const double b = 0.0;
    const double c = 1.0;
    return a*std::exp(-((x-b)*(x-b))/(2*c*c));
}
```



Taylor's Formula

Meaning



$$f(x) = f(a) + (x-a)f'(a)$$

$$f(x) = c_0 + c_1 x + c_2 x^2$$

$$f(x)=c_0+c_1x+c_2x^2+c_3x^3+\dots$$

Definition

$$f(x)=f(a)+\frac{f'(a)}{1!}(x-a)+\frac{f''(a)}{2!}(x-a)^2+\frac{f'''(a)}{3!}(x-a)^3+\dots$$
$$+\frac{f^k(a)}{k!}(x-a)^k+h_k(x)(x-a)^k$$
$$\lim_{x\to a}h_k(x)=0$$

Peano form of the remainder

$$\frac{f'''(a)}{3!}(x-a)^3$$
$$\frac{3\bullet f'''(a)}{3!}(x-a)^2$$
$$\frac{3\bullet 2\bullet f'''(a)}{3!}(x-a)^1$$
$$\frac{3\bullet 2\bullet 1\bullet f'''(a)}{3!}(x-a)$$

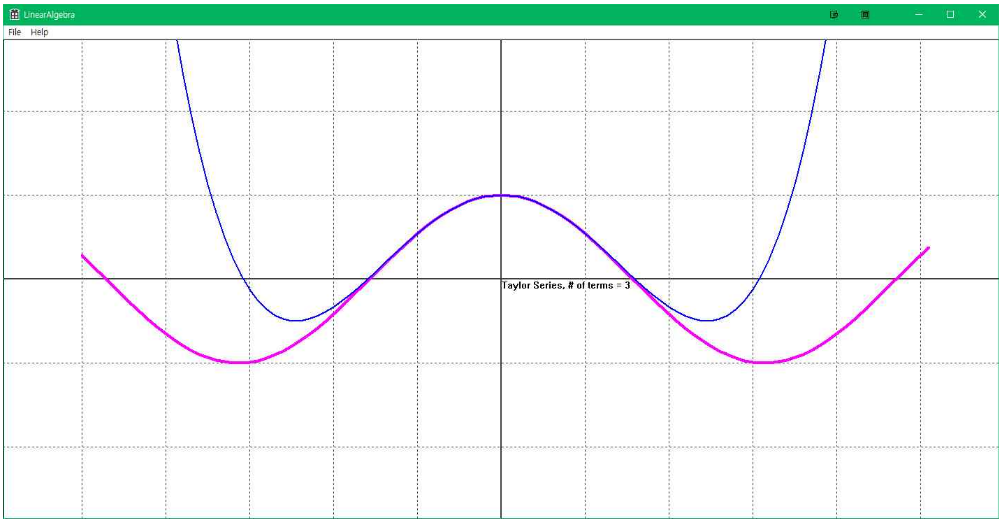
Examples

$$\sin(0)=0$$
$$\sin'(0)=\cos(0)=1$$
$$\cos'(0)=-\sin(0)=0$$
$$-\sin'(0)=-\cos(0)=-1$$
$$-\cos'(0)=--\sin(0)=0$$
$$\sin'(0)=\cos(0)=1$$
$$\dots$$
$$\sin(x)=\sum_{n=0}^{\infty}\frac{(-1)^n}{(2n+1)!}x^{2n+1}=x-\frac{x^3}{3!}+\frac{x^5}{5!}+\dots$$

$$\cos(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} + \dots$$

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots$$

Implementation




```
double Cosine(double x)
{
	return cos(x);
}

long long factorial(long long x, long long result = 1)
{
	if (x == 0)
		return result;
	else
		return factorial(x - 1, x * result);
}
```

```
long long numberOfTaylorSeriesTerms = 3;
double TaylorCosine(double x)
{
    double result = 0;
    for (int n = 0; n < numberOfTaylorSeriesTerms; ++n)
    {
        result += (std::pow(-1, n) / factorial(2 * n))*std::pow(x, 2 * n);
    }
    return result;
    //return 1 - (x*x) / (2*1) + (x*x*x*x) / (4 * 3 * 2 * 1);
}
```

Proof

$$f'(a) = \lim_{x \to a} \frac{f(x) - f(a)}{x - a}$$

$$f'(a) = \frac{f(x) - f(a)}{x - a}$$

$$f(x) = f(a) + (x - a) f'(a)$$

$$f(x) - f(a) = \int_a^x f'(\xi) d\xi$$

$$f(x) = f(a) + \int_a^x f'(\xi) d\xi$$

$$f'(x) = f'(a) + (x - a) f''(a)$$

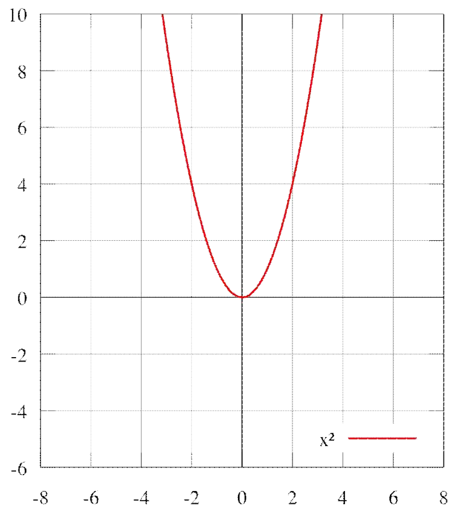
$$f(x) = f(a) + \int_a^x \{f'(a) + (x-a)f''(a)\} d\xi$$
$$= f(a) + f'(a)[\xi]_a^x + f''(a)[\frac{1}{2}(\xi - a)^2]_a^x$$
$$= f(a) + (x-a)f'(a) + \frac{1}{2}(x-a)^2 f''(a)$$

...

Euler's Formula

Even function

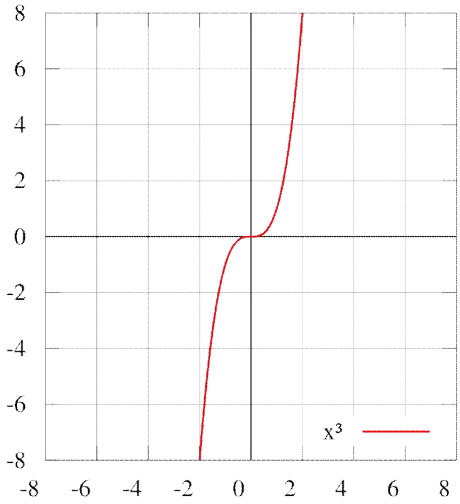
$f(x) = f(-x)$



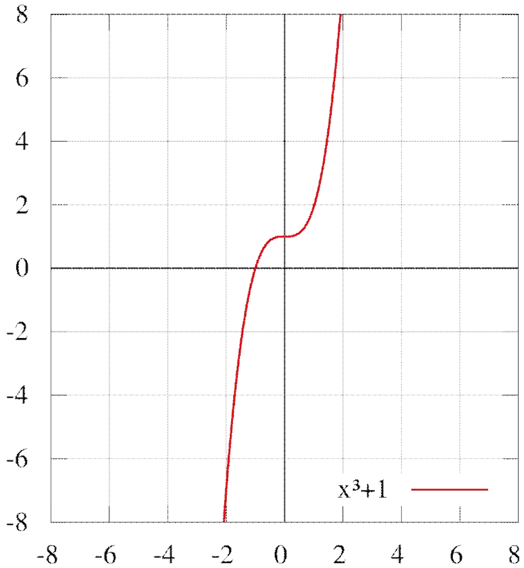
$f(x) = x^2$

Odd function

$-f(x) = f(-x)$



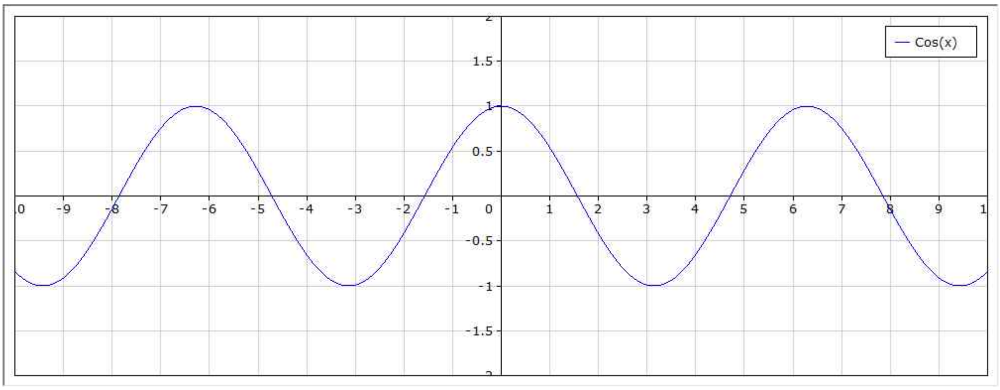
$$f(x) = x^3$$



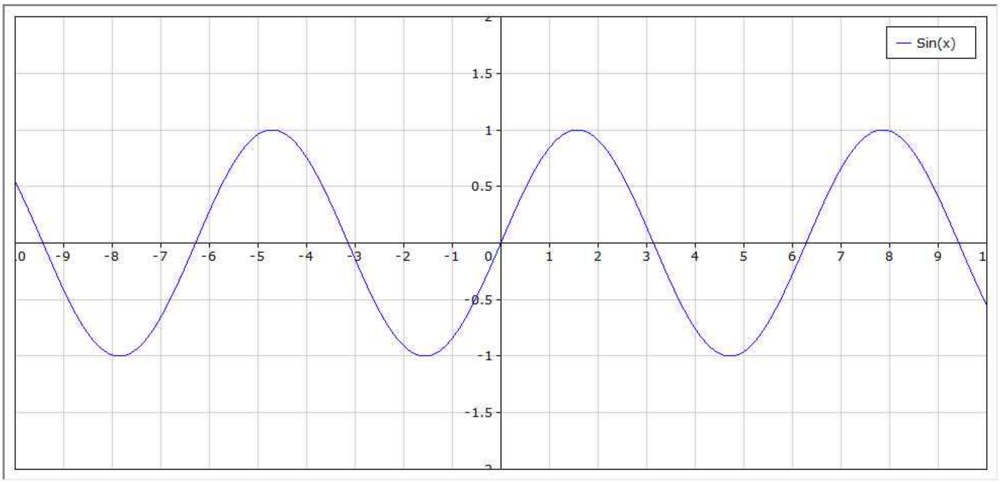
$$f(x) = x^3 + 1$$

not even or odd

Cosine and Sine



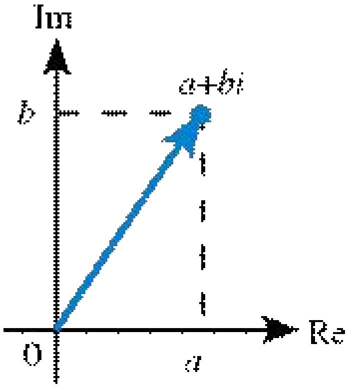
$\cos(x)$



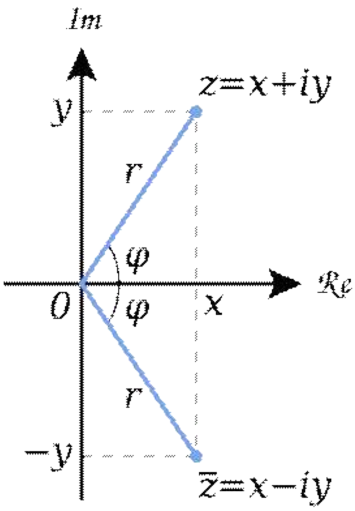
$\sin(x)$

Complex number

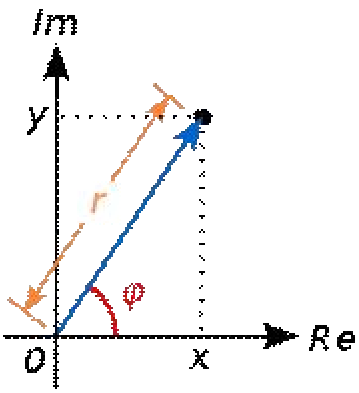
A complex number is a number that can be expressed in the form $a+bi$, where a and b are real numbers, and i is a solution of the equation $x^2=-1$.



Conjugate

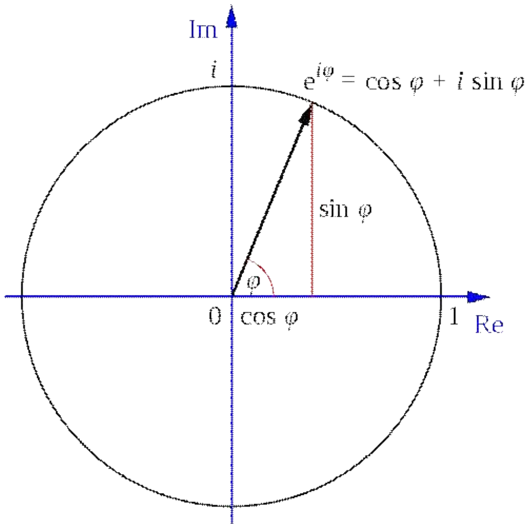


Polar Coordinate



$r(\cos(\theta) + i\sin(\theta))$

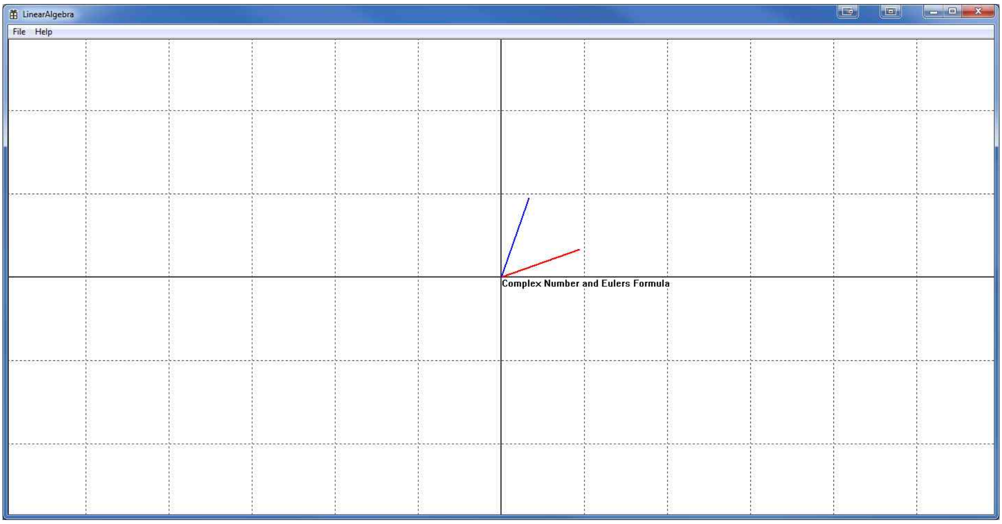
Euler's formula



$$e^{ix} = \cos(x) + i\sin(x)$$

$$(\cos(x) + i\sin(x))(\cos(y) + i\sin(y)) = ?$$
$$e^{ix}e^{iy} = e^{i(x+y)} = \cos(x+y) + i\sin(x+y)$$

Implementation




```
    static double timer = 0;
    timer += (double)fElapsedTime_;
    const std::complex<double> i( 0, 1 );
    std::complex<double>    c0;
    c0 = std::polar<double>( 1.0, M_PI / 4.0 );

    {
        std::complex<double>    c1;
        c1 = std::polar<double>( 1.0, timer );
        double theta = std::arg( c0 * c1 );

        KMatrix2 m;
        m.SetRotation( theta );
        KVector2 v( 1, 0 );
        v = m * v;
        KVectorUtil::DrawLine( hdc, KVector2::zero, v, 2, PS_SOLID,
RGB( 255, 0, 0 ) );
    }

    {
        std::complex<double>    c2;
        c2 = std::exp( i * -timer );
        double theta = std::arg( c0 * c2 );

        KMatrix2 m;
```

```
        m.SetRotation( theta );
        KVector2 v( 1, 0 );
        v = m * v;
        KVectorUtil::DrawLine( hdc, KVector2::zero, v, 2, PS_SOLID,
RGB( 0, 0, 255 ) );
    }
```

Conjugate

$$e^{-ix} = \cos(-x) + i\sin(-x) = \cos(x) - isin(x)$$

$$e^{ix}e^{-ix} = e^{i(x-x)} = e^{0i} = e^{0} = 1$$

Pythagorean theorem

$$e^{ix}e^{-ix} = (\cos(x) + isin(x))(\cos(x) - isin(x))$$
$$= \cos^2(x) - \cos(x)i\sin(x) + isin(x)\cos(x) - i^2\sin^2(x)$$
$$= \cos^2(x) + \sin^2(x) = 1$$

Proof

$$e^{x} = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$$

$$\sin(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \ldots$$

$$\cos(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} + \ldots$$

$$e^{ix} = \sum_{n=0}^{\infty} \frac{(ix)^n}{n!} = 1 + ix + \frac{(ix)^2}{2!} + \frac{(ix)^3}{3!} + \ldots$$

$$e^{ix} = \cos(x) + i\sin(x)$$

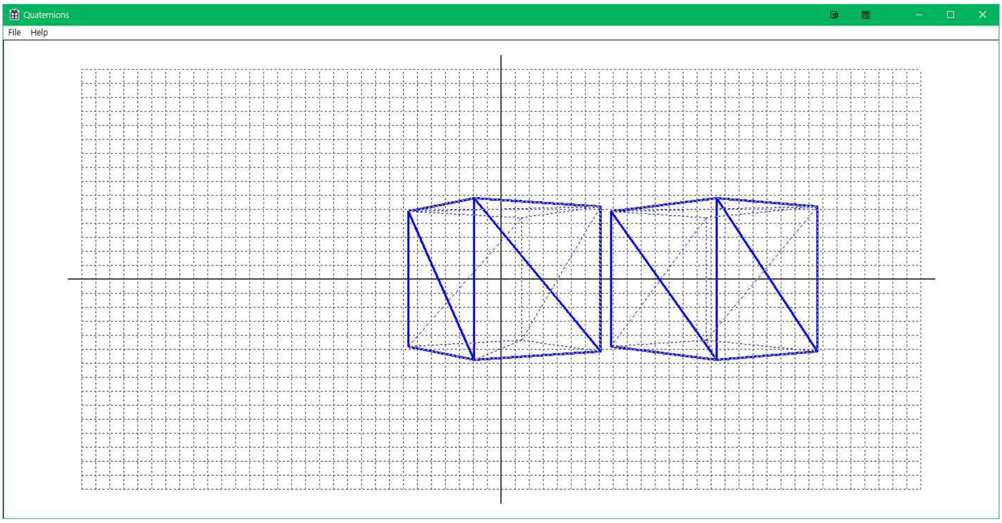
Quaternion

$$e^{w+xi+yj+zk} = e^{a}\left(\cos(|v|) + \frac{\sin(|v|)}{|v|}(xi+yj+zk)\right)$$

$$|v| = \sqrt{x^2+y^2+z^2}$$

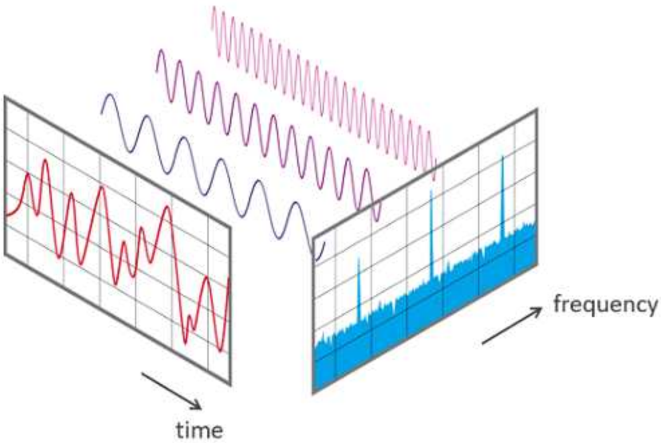
$$e^{(w+xi+yj+zk)}e^{(ai+bj+ck)} = e^{w}e^{(x+a)i+(y+b)j+(z+c)k}$$

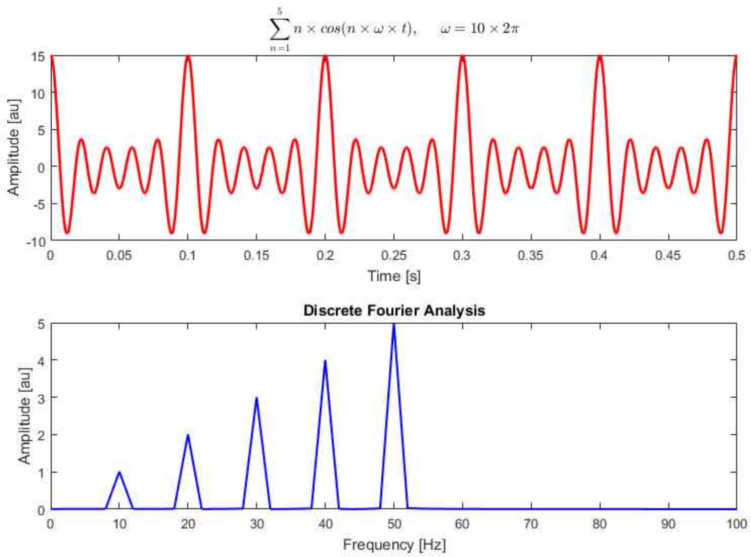
$$e^{0}e^{(x+a)i+(y+b)j+(z+c)k} = e^{(x+a)i+(y+b)j+(z+c)k}$$



Fourier Transform

Observation

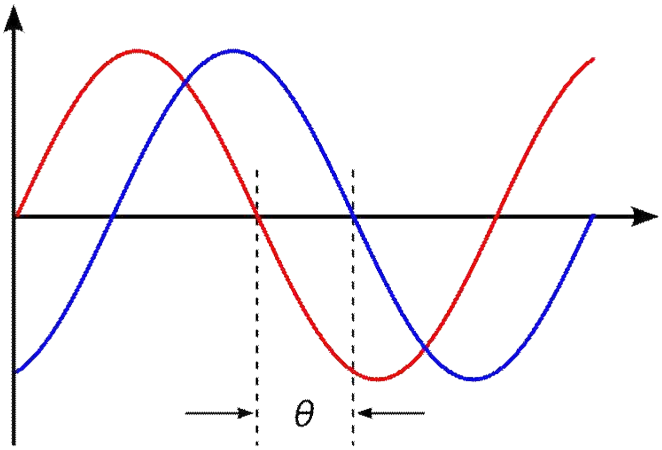




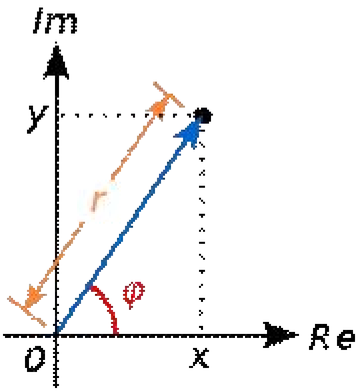
$$f(v) = \cos(2\pi t) + 0.5\cos(2\pi 4t) + \dots$$

$$f(t) = \sum_{v=-\infty}^{\infty} A(v)\cos(2\pi v t)dv$$

Phase shift



$$f(v) = \cos(2\pi t) + 0.5\cos(2\pi 4t + \pi/4) + \ldots$$

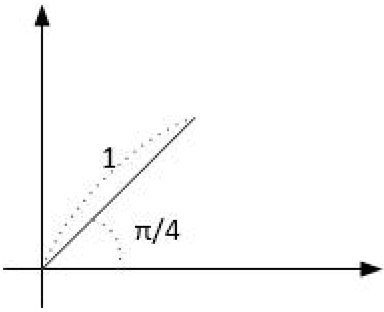


$$F(v) = r(\cos(\theta) + i\sin(\theta))$$

$$e^{ix} = \cos(x) + i\sin(x)$$

$$f(t) = \sum_{v=-\infty}^{\infty} F(v)e^{(2\pi ivt)}dv$$

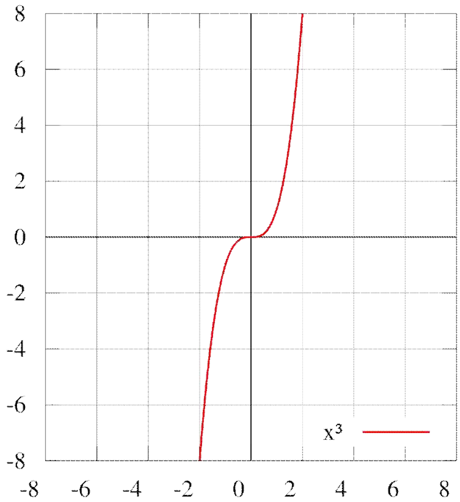
Example



$$
\begin{aligned}
&(\cos(\pi/4)+i\sin(\pi/4))e^{2\pi i v t} \\
&=(\cos(\pi/4)+i\sin(\pi/4))(\cos(2\pi v t)+i\sin(2\pi v t)) \\
&=\cos(\pi/4)\cos(2\pi v t)-\cos(\pi/4)\sin(2\pi v t) \\
&+\cos(\pi/4)\cos(2\pi v t)i-\cos(\pi/4)\sin(2\pi v t)i
\end{aligned}
$$

Odd function

$$-f(x)=f(-x)$$



Inverse Fourier Transform

$$f(t) = \sum_{v=-\infty}^{\infty} F(v)e^{2\pi ivt}dv$$

Forward Fourier Transform

$$F(v) = \sum_{v=-\infty}^{\infty} f(t)e^{-2\pi ivt}dt$$

Hint

$$\frac{f(t)}{e^{2\pi ivt}}$$

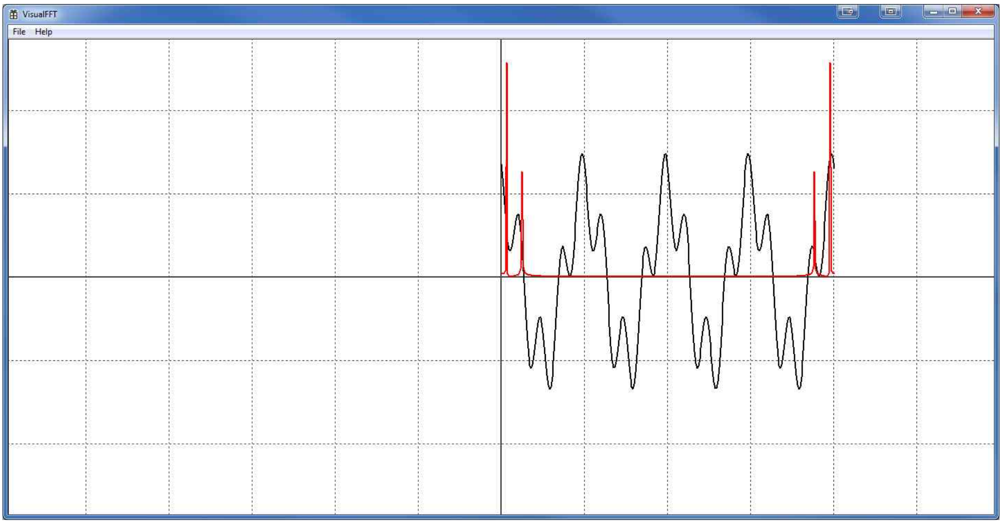
Implementation

```
using ComplexArray = std::valarray<std::complex<double> >;

void dft(ComplexArray& x)
{
    const size_t N = x.size();

    ComplexArray xresult(N);

    for (size_t j = 0; j < N; ++j)
    {
        xresult[j] = 0;
        for (size_t k = 0; k < N; ++k)
        {
            std::complex<double> t = std::polar(1.0, -2 * M_PI * j * k / N);
            xresult[j] += x[k] * t;
        }
    }
    x = xresult;
}
```



Fast Fourier Transform

See wiki ^^;

@