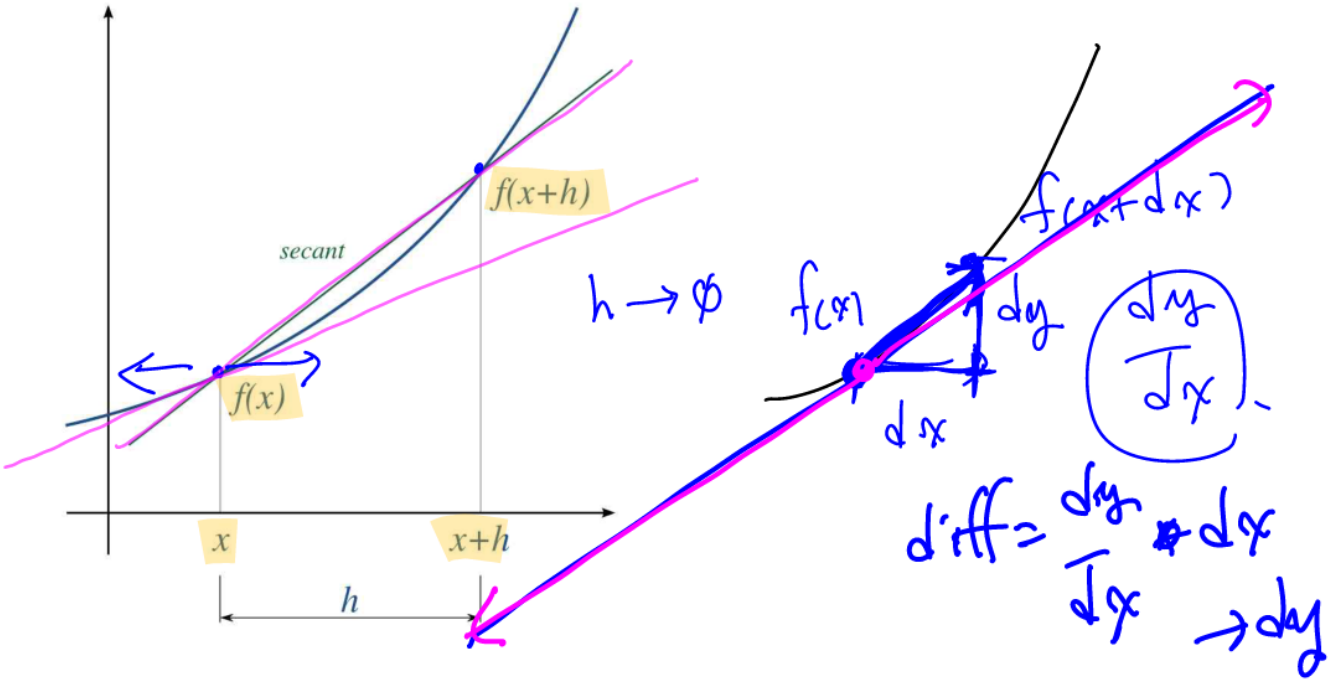
Numerical Differentiation



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

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

error $\leq h^1$

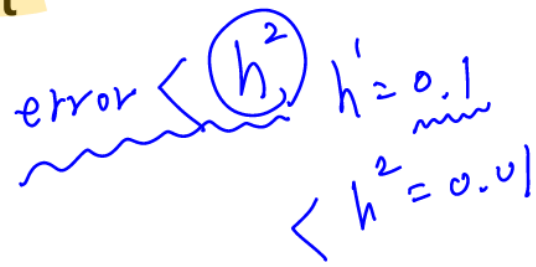
derivative of f at x

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

0.0001

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

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

4th order error < h^4

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

```
using FUNCTION = double(*)( double x );

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

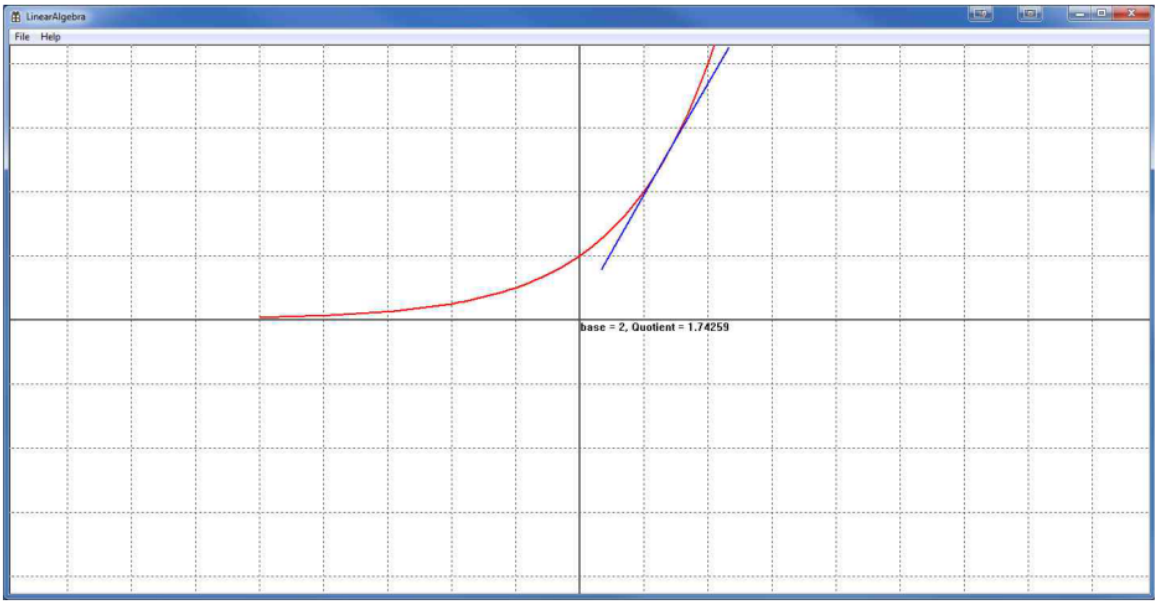
Drawing a Tangent Line Segment

```
	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