Complex Arithmetic

```
> db.complex.find().pretty()
{
    "_id" : ObjectId("53bf040a66f35625b75d3eb3"),
    "a" : {
        "Complex" : {
            "re" : 2,
            "im" : 3
        }
    },
    "b" : {
        "Complex" : {
            "re" : 4,
            "im" : 5
        }
    }
}
```

```
function Subtract() {
    return [
        { "$project": {
            _id: 0,
            a  : 1,
            b  : 1,
            difference: {
                Complex: {
                    re: {
                        "$subtract": [
                            Re( "a" ),
                            Re( "b" )
                        ]
                    },
                    im: {
                        "$subtract": [
                            Im( "a" ),
                            Im( "b" )
                        ]
                    }
                }
            }
        } }
    ];
}
```

```
function Re( c ) {
    return "$" + c + ".Complex.re";
}

function Im( c ) {
    return "$" + c + ".Complex.im";
}

function Add() {
    return [
        { "$project": {
            _id: 0,
            a  : 1,
            b  : 1,
            sum: {
                Complex: {
                    re: {
                        "$add": [
                            Re( "a" ),
                            Re( "b" )
                        ]
                    },
                    im: {
                        "$add": [
                            Im( "a" ),
                            Im( "b" )
                        ]
                    }
                }
            }
        } }
    ];
}
```

```
function Multiply() {
    return [
        { "$project": {
            _id: 0,
            a  : 1,
            b  : 1,
            product: {
                Complex: {
                    re: {
                        "$subtract": [
                            {
                                "$multiply": [
                                    Re( "a" ),
                                    Re( "b" )
                                ]
                            },
                            {
                                "$multiply": [
                                    Im( "a" ),
                                    Im( "b" ),
                                ]
                            }
                        ]
                    },
                    im: {
                        "$add": [
                            {
                                "$multiply": [
                                    Re( "a" ),
                                    Im( "b" )
                                ]
                            },
                            {
                                "$multiply": [
                                    Im( "a" ),
                                    Re( "b" )
                                ]
                            }
                        ]
                    }
                }
            }
        } }
    ];
}
```

Equation Solving

$ax + b = 0$

```
db.linear.find().pretty()
{
    "_id" : ObjectId("53c011ca66f35625b75d3eb5"),
    "linear" : {
        "a" : 2,
        "b" : -6
    }
}
```

```
function Linear() {
    return [
        { "$project": {
            _id: 0,
            a: "$linear.a",
            b: "$linear.b",
            min_b: {
                "$subtract": [ 0, "$linear.b" ]
            }
        } },
        { "$project": {
            x: {
                "$divide" : [ "$min_b", "$a" ]
            }
        } }
    ];
}
```

```
{ "result" : [ { "x" : 3 } ], "ok" : 1 }
```

$ax^2 + bx + c = 0$

```
> db.quadratic.find().pretty()
{
    "_id" : ObjectId("53c0188a66f35625b75d3eb6"),
    "quadratic" : {
        "a" : 3,
        "b" : 2,
        "c" : -33
    }
}
```

```
function Quadratic() {
    var pipe = [
        { $project : {
            _id:   0,
            a: "$quadratic.a",
            b: "$quadratic.b",
            c: "$quadratic.c",
            min_b: { "$subtract" : [ 0, "$quadratic.b" ] },
            two_a: { "$multiply" : [ 2, "$quadratic.a" ] },
            four_a_c: { "$multiply" : [ 4, "$quadratic.a", "$quadratic.c" ] },
            b_sq: { "$multiply" : [ "$quadratic.b", "$quadratic.b" ] }
        } },
        { $project: {
            min_b: 1,
            two_a: 1,
            n: { "$subtract": [ "$b_sq", "$four_a_c" ] }
        } },
    ];

    pipe = pipe.concat(Newton());

    rest = [
        { $project: {
            x1: { "$divide": [ { "$add":      [ "$min_b", "$r" ] }, "$two_a" ] },
            x2: { "$divide": [ { "$subtract": [ "$min_b", "$r" ] }, "$two_a" ] }
        } }
    ];

    pipe = pipe.concat( rest );

    return pipe;
}
```

```
{
    "result" : [ {
            "x1" : 3.000000000049738,
            "x2" : -3.6666666667164045
        } ],
    "ok" : 1
}
```

Calculus

```
function get_sized_derive_pipeline(targetfield, outfield, degree) {
    //generate projection to fill in null coefficients with 0
    var nullproj = {$project: {_id: 1}};
    var newx = {};
    for(var i = 0; i <= degree; i++) {
        newx[i] = {$ifNull: ["$" + targetfield + "." + i, 0]};
    }
    nullproj["$project"][targetfield] = newx;

    //generate projection to do actual derivation
    var deriveproj = {$project: {_id: 1}};
    var xprime = {};
    for(var i = 0; i < degree; i++) {
        xprime[i] = {$multiply: ["$" + targetfield + "." + (1 + i), (1 + i)]};
    }
    deriveproj["$project"][outfield] = xprime;

    //return pipeline
    return [nullproj, deriveproj];
}
```

```
function get_sized_integrate_pipeline(targetfield, outfield, degree) {
    //generate projection to fill in null coefficients with 0
    var nullproj = {$project: {_id: 1}};
    var newx = {};
    for(var i = 0; i <= degree + 1; i++) {
        newx[i] = {$ifNull: ["$" + targetfield + "." + i, 0]};
    }
    nullproj["$project"][targetfield] = newx;

    //generate projection to do actual derivation
    var integrateproj = {$project: {_id: 1}};
    var xint = {};
    for(var i = 0; i <= degree + 1; i++) {
        if(i === 0) {
            xint[i] = {$multiply: [1, 0]};//need to set it to 0 somehow lol
        } else {
            xint[i] = {$divide: ["$" + targetfield + "." + (i - 1), i]};
        }
    }
    integrateproj["$project"][outfield] = xint;

    //return pipeline
    return [nullproj, integrateproj];
}
```

```
Initial polynomial:
{
    "_id" : 0,
    "x" : {
        "0" : 1,
        "1" : 2,
        "2" : 3,
        "3" : 4,
        "4" : 5,
        "5" : 6
    }
}
```

```
[
    {
        "_id" : 0,
        "y" : {
            "0" : 0,
            "1" : 1,
            "2" : 1,
            "3" : 1,
            "4" : 1,
            "5" : 1,
            "6" : 1
        }
    }
]
```

```
[
    {
        "_id" : 0,
        "y" : {
            "0" : 2,
            "1" : 6,
            "2" : 12,
            "3" : 20,
            "4" : 30
        }
    }
]
```

http://github.com/friedo/agg-symbolic