Recursive vectors (Part 2)

Gonzalo Reynaga García
March 2019

This is an extension of the recursive vector article, it discusses about the magnitude of the vector and the calculation of the inverse vector using iteration method.

Magnitude

The magnitude of the recursive vector is calculated the same as the magnitude of the complex number. It is the sum square of each element; this definition will be used for the magnitude of the vector.

$$\|V\| = \sqrt{\sum_{k=0}^{N-1} v_k^2}$$

When multiply a vector V with vector U, the result vector magnitude is not the multiplication of each vector magnitude.

$$\|V \hat{} U\| \neq \|V\| \cdot \|U\|$$

If V and U has magnitude 1, the magnitude after the operation could be from 0 to 2, the exact range has not been determined.

To test the magnitude of the vector, the geometric succession is used.

$$1 + r + r^2 + r^3 + r^4 + \cdots = \sum_{k}^{\infty} r^k$$

When r is a scalar or complex number the magnitude must be less than 1 in order to the sum converge. But with recursive vectors the magnitude must be much less than 1, and the numeric result suggest that a magnitude of less or equal of 0.5 ensure the series converge.

Math-c script used

```
////////////////////////////////////////////////
//vectorize an array of numbers WITHOUT
//imaginary part(scalars were not merged)
//example: vectorize([3 4]) = 3+4i

function vectorize(a)
            vSize = max(size(a))(0);

            Ra = a(0:(vSize/2-1));
            Ia = a((vSize/2):vSize-1);

            if vSize == 2
                        if isArray(a(0)) == 0
                                    return a(0) + 1i*a(1);
                        end
                        Ra = Ra(0);
                        Ia = Ia(0);
            end

            Ra = vectorize(Ra);
            Ia = vectorize(Ia);
            return [Ra Ia];
end

////////////////////////////////////////////////
//remove the brackets to convert it to an array of numbers.
function unbracket(a)
            vSize = max(size(a))(0);
            if max(size(a(0)))(0) == 1
                        return [real(a(0)) imag(a(0)) real(a(1)) imag(a(1))];
            end
            vectorArray = [];
            for i=0:vSize-1
                        vectorArray = vectorArray <-> unbracket(a(i));
            end
            return vectorArray;
end


function magV(v)
            v = unbracket(v);
            v = sumsq(v);
            return sqrt(v);
end


clc();
clear();
//
V=[1 9 6 5 4 5 3 3];

vecsize = max(size(V))(0);
one = zeros(1,vecsize);
one(0) = 1;

one = vectorize(one);
V = vectorize(V);

m = magV(V)*2
r = V/m
magV(r)
s = one;
for i=1:10
            s=s^r + one
end
```

Output

```
m = 28.4253
r = [[0.0351799+0.316619i  0.211079+0.175899i]  [0.14072+0.175899i  0.10554+0.10554i]]
ans = 0.5
s = [[1.03518+0.316619i  0.211079+0.175899i]  [0.14072+0.175899i  0.10554+0.10554i]]
s = [[0.933695+0.237411i  0.12197+0.255107i]  [0.0318086+0.195701i  0.0436584+0.303559i]]
s = [[0.979324+0.232186i  0.143392+0.187534i]  [0.0870166+0.103049i  -0.0765104+0.252357i]]
s = [[0.934831+0.242185i  0.168389+0.231157i]  [0.1326+0.161377i  -0.016308+0.201186i]]
s = [[0.948096+0.199572i  0.128179+0.220535i]  [0.0879738+0.164412i  -0.00853299+0.245672i]]
s = [[0.971614+0.224526i  0.152184+0.197531i]  [0.104112+0.141333i  -0.0301908+0.230824i]]
s = [[0.948377+0.23004i  0.157184+0.220357i]  [0.107905+0.160117i  -0.0115128+0.227513i]]
s = [[0.953865+0.21549i  0.142629+0.214618i]  [0.0952082+0.153144i  -0.0186566+0.240163i]]
s = [[0.959198+0.224283i  0.151487+0.209254i]  [0.104964+0.148525i  -0.0232907+0.230317i]]
s = [[0.952153+0.223261i  0.150479+0.216323i]  [0.103719+0.156151i  -0.0156396+0.231539i]]
```

Division (inverse)

In the first part we defined the analytic inverse as:
The inverse of recursive vector V:

$$V = [Re(V) \quad Im(V)]$$

$$\frac{1}{V} = [Re(V) \quad -Im(V)] \cdot \left(\frac{1}{[Re(V) \cdot Re(V) \quad Im(V) \cdot Im(V)]}\right)$$

Due a floating-point problem when using large vectors, a analytic solution is not a good option. Then a division algorithm was used below.
Based in the reference [2] and [3] we tried to implement the division algorithm in the recursive vector.

The first step is set the vector to magnitude 0.5 and use the scalar magnitude separately.

$$\frac{1}{V} = \frac{1}{2 \cdot \|V\| \cdot \vec{V}}$$

Where:
$\overrightarrow{V}$ is the vector of magnitude 0.5

Then we apply the division algorithm

$$X_{i+1} = X_i \text{^} \left([2] - \vec{V} \text{^} X_i\right)$$

With enough iterations

$$\frac{1}{V} \approx \frac{X_{i+1}}{2 \cdot \|V\|}$$

One important point is the initial value of X_0, not all the initial values make the division algorithm converge to the solution. With the numeric tests was found that using

$$X_0 = Conjugate(\vec{V})$$

The serie converges.

The Math-c script used to implement the division algorithm.

```
function vectorize(a)
            vSize = max(size(a))(0);

            Ra = a(0:(vSize/2-1));
            Ia = a((vSize/2):vSize-1);

            if vSize == 2
                        if isArray(a(0)) == 0
                                    return a(0) + 1i*a(1);
                        end
                        Ra = Ra(0);
                        Ia = Ia(0);
            end

            Ra = vectorize(Ra);
            Ia = vectorize(Ia);
            return [Ra Ia];
end

////////////////////////////////////////////////////
//remove the brackets to convert it to an array of numbers.
function unbracket(a)
            vSize = max(size(a))(0);
            if max(size(a(0)))(0) == 1
                        return [real(a(0)) imag(a(0)) real(a(1)) imag(a(1))];
            end
            vectorArray = [];
            for i=0:vSize-1
                        vectorArray = vectorArray <-> unbracket(a(i));
            end
            return vectorArray;
end

//////////////////////////////////////
// Recursive conjugate of vector
function cjV(v)
            vSize = max(size(v))(0);
            if(min(size(vSize))(0) != 1)
                        print("cjv -> error\n");
                        return 0;
            end
            if(vSize == 1 )
                        return v';
            end
            Rr = cjV(v(0));
            Ri = -cjV(v(1));
            return [Rr Ri];
end

function magV(v)
            v = unbracket(v);
            v = sumsq(v);
            return sqrt(v);
end
//////////////////////////////////////////
//Iteration division of recursive vector
function invV(V)
            vecsize = max(size(unbracket(V)))(0);
            two = 2 <-> ((2: vecsize)*0);
            two = vectorize(two);
            m = magV(V)*2;
            Vn = V/m;
            X = cjV(Vn);
            for i=0:20
                        X=X^(two-(Vn^X));
            end
            return X/m;
end

clc();
clear();

////////////////////////////
//singulars vectors [[0.5  0]  [0.5i  0]] ----- [[1 0] [1i 0]]
//Note: vecsize must be power of 2

vecsize = 8;
V=vectorize(randi(10,1, vecsize));


print("Iteration inverse\n");
invV(V)
print("Analytic inverse\n");
inv(V)
```

Reference

[1] https://en.wikipedia.org/wiki/Geometric_series
[2] https://en.wikipedia.org/wiki/Division_algorithm
[3] https://www.youtube.com/watch?v=AoVI9NWegWw