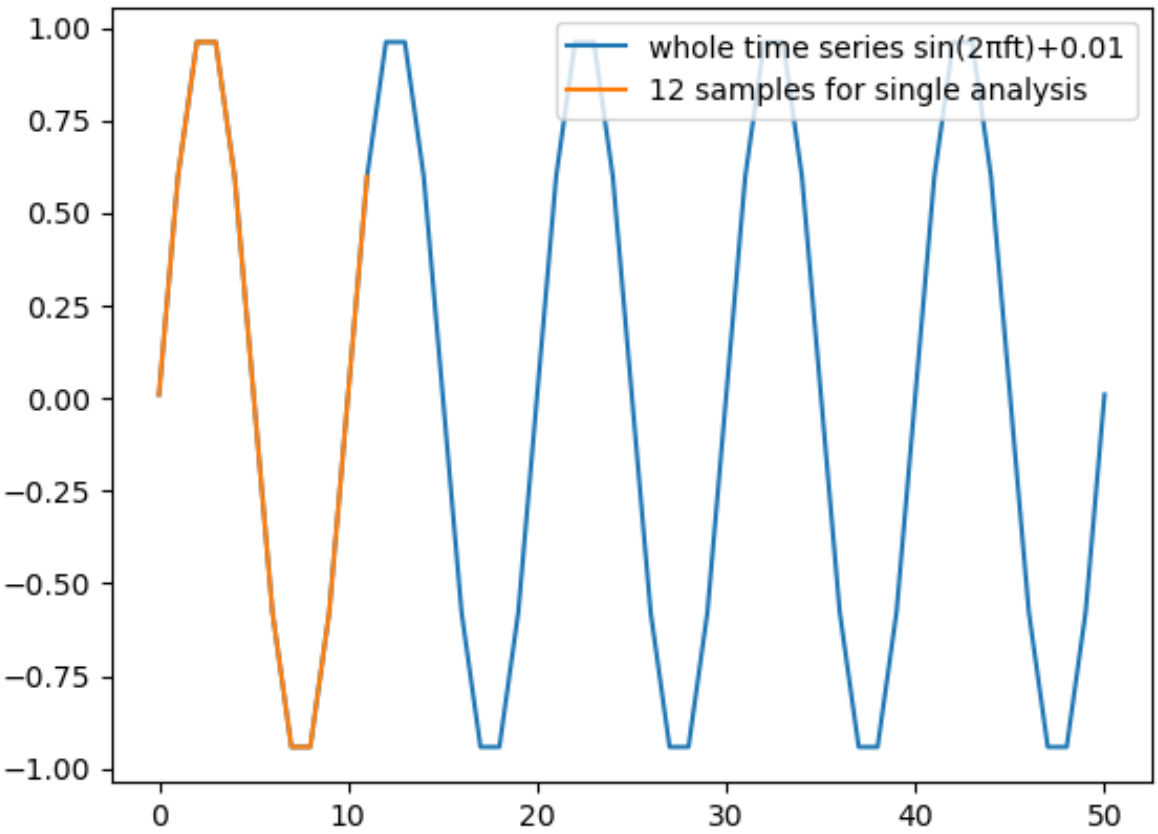
```
julia> #=====        ***** tutorial to show how it works *****        =====#
julia> using PyCall
julia> using PyPlot
julia> using LinearAlgebra
julia> using Polynomials
julia> #ts=sinpi.(range(0,2,length=TotN));
julia> #ts=exp2.(0:0.25:TotN);
julia> ts=sinpi.(0:0.2:10) .+0.01;
julia> M=7; N=5;
julia> TotN=N+M;
julia> # N > ramk(M) will be reqruired to obtain stable solution
julia> #        to avoid singularity by degeneration
julia> figure();
julia> plot(ts, label="whole time series sin(2πft)+0.01");
julia> plot(ts[1:TotN], label="$TotN samples for single analysis");
julia> legend(); show();
```

解析対象波形 ts=sin πt +0.01, t=0, 0.2, 0.4, ..., 10
正弦波＋定数としていることから、必要な解析次数はM=3となる。
次の行で数値計算上の解析次数をM=7としているので、計算途中で解析次数が4つ下がることになる。
この、解析次数が下がることにともなう様々な例外処理が必要になることから、それらをこのプログラム事例を通して説明する。

経験上、TotN>2Mとしておくのがよさそう

一回の解析に用いる区間: ここでは12サンプル
このコード上では、ts[1:TotN]のみを示しているが、実際にはts[1+k:TotN+k], k=0,1,2,...として解析に用いる区間をずらしながら計算を進めていく。



参考としてJuliaでの関数の書き方を例示

```
julia> #-------------------------------- preset functions; julia exhiibitions
julia> """find extreme value in matrix A""";
julia> extreme(Aij)=
            Aij |> maximum |> abs > Aij |> minimum |> abs ? maximum(Aij) : minimum(Aij);
julia> """find exterme value in vector x""";
julia> absmax(x)=maximum(abs.(extrema(x)));
julia> """corresponding index of above vector x""";
julia> absmaxidx(x)=filter(i-> abs(x[i]) == absmax(x), eachindex(x));
```

```
julia> myeps=1e-15;
julia> #---------------------------- set autocorrelation eq. AX=B
julia> A=zeros(M,M);
julia> for m in 1:M, m′   in 1:M, n in 0:N-1
            A[m,m′  ]+=ts[TotN-m-n]*ts[TotN-m′  -n]
       end
julia> A
7×7 Matrix{Float64}:
  2.43895    1.96149    0.723245  -0.802819  -2.0338    -2.4995    -2.02204
  1.96149    2.43895    1.97324    0.742266  -0.783798  -2.02204   -2.4995
  0.723245   1.97324    2.46246    2.00402    0.773042  -0.760287  -2.01029
 -0.802819   0.742266   2.00402    2.5005     2.04206    0.803819  -0.741266
 -2.0338    -0.783798   0.773042   2.04206    2.53854    2.07284    0.82284
 -2.4995    -2.02204   -0.760287   0.803819   2.07284    2.56205    2.0846
 -2.02204   -2.4995    -2.01029   -0.741266   0.82284    2.0846     2.56205
julia> B=zeros(M);
julia> for m′   in 1:M, n in 0:N-1
            B[m′  ] += ts[TotN-0-n]*ts[TotN-m′  -n]
       end
julia> B'
1×7 adjoint(::Vector{Float64}) with eltype Float64:
 1.97324   0.723245  -0.810085  -2.04106  -2.4995  -2.01029  -0.760287
julia> #----------------------------------- normalize matrix
julia> scaler=extreme(A);
julia> A /=scaler;
julia> B /=scaler;
julia> A
7×7 Matrix{Float64}:
   0.95195    0.765592   0.282291  -0.31335   -0.793816  -0.975585  -0.789227
   0.765592   0.95195    0.770181   0.289715  -0.305926  -0.789227  -0.975585
   0.282291   0.770181   0.961127   0.782193   0.301728  -0.296749  -0.784639
  -0.31335    0.289715   0.782193   0.975975   0.797042   0.31374   -0.289325
  -0.793816  -0.305926   0.301728   0.797042   0.990823   0.809054   0.321164
  -0.975585  -0.789227  -0.296749   0.31374    0.809054   1.0        0.813643
  -0.789227  -0.975585  -0.784639  -0.289325   0.321164   0.813643   1.0
julia> B'
1×7 adjoint(::Vector{Float64}) with eltype Float64:
 0.770181   0.282291  -0.316186  -0.796651  -0.975585  -0.784639  -0.296749
julia> #----------------- care for void records: safety reason
julia> for i in 1:M
            if norm(A[i,:]) < myeps
                A[i,:]=zeros(M)
                B[i]=0
                A[i,i]=1
            end
```

数値計算途上でのランク低下に追従できないよう、問題点が残るよう、わざと値を小さくしてある。

教科書の定義そのものの計算式

自己相関行列：どの教科書でもテプリッツ行列になると説明されているが、第二章で示した通り、対角項の値が同一値になっていないことで、テプリッツ行列になっていないことを確認できる。

自己相関行列中の最大要素を用いてスケーリング
生データを用いて行列を作った場合のレンジを調整

行列中の最大値が1

理論解析を行う場合、行列の行内の値が全て0となる場合も発生しうる。それに備えての例外処理。以下でも繰り返し同じ処理が出てくる。数値計算上0になり切れず残差が発生する場合への手当にもなっている。

```
        end
julia> A
```
例外処理の対象行が無いので不変

```
7×7 Matrix{Float64}:
   0.95195    0.765592   0.282291  -0.31335   -0.793816  -0.975585  -0.789227
   0.765592   0.95195    0.770181   0.289715  -0.305926  -0.789227  -0.975585
   0.282291   0.770181   0.961127   0.782193   0.301728  -0.296749  -0.784639
  -0.31335    0.289715   0.782193   0.975975   0.797042   0.31374   -0.289325
  -0.793816  -0.305926   0.301728   0.797042   0.990823   0.809054   0.321164
  -0.975585  -0.789227  -0.296749   0.31374    0.809054   1.0        0.813643
  -0.789227  -0.975585  -0.784639  -0.289325   0.321164   0.813643   1.0
julia> B'
1×7 adjoint(::Vector{Float64}) with eltype Float64:
 0.770181  0.282291  -0.316186  -0.796651  -0.975585  -0.784639  -0.296749
julia> #---------------------------------- pivotting
julia> for i in 1:M-1
            amidx=absmaxidx(A[i:M,i])[1]+i-1
            A[amidx,:], A[i,:] = A[i,:], A[amidx,:]
            B[amidx], B[i] = B[i], B[amidx]
        end
julia> A
```
掃き出し法を用いるための準備。予測係数を求めることが目的なので、上側にある行（低い次数に対応）ほど重要。下側にある行（高い次数に対応）はできるだけゼロ行列になるようにして、全体での解析次数をできるだけ下げられるようにする必要がある。一般の行列の対角化では、このようなプライオリティ関係が存在しないので、別途補う必要がある。

```
7×7 Matrix{Float64}:
  -0.975585  -0.789227  -0.296749   0.31374    0.809054   1.0        0.813643
  -0.789227  -0.975585  -0.784639  -0.289325   0.321164   0.813643   1.0
   0.282291   0.770181   0.961127   0.782193   0.301728  -0.296749  -0.784639
  -0.31335    0.289715   0.782193   0.975975   0.797042   0.31374   -0.289325
  -0.793816  -0.305926   0.301728   0.797042   0.990823   0.809054   0.321164
   0.95195    0.765592   0.282291  -0.31335   -0.793816  -0.975585  -0.789227
   0.765592   0.95195    0.770181   0.289715  -0.305926  -0.789227  -0.975585
julia> B'
1×7 adjoint(::Vector{Float64}) with eltype Float64:
 -0.784639  -0.296749  -0.316186  -0.796651  -0.975585  0.770181  0.282291
julia> #---------------------------------- sweepout forward
julia> for i in 1:M-1
            if abs(A[i,i]) < myeps
                A[i,:]=zeros(M)
                B[i]=0
                A[i,i]=1
            end
            for j=i+1:M
                mx=A[j,i]/A[i,i]
                A[j,:].-=mx*A[i,:]
                B[j] -= mx*B[i]
            end
        end
julia> A
```
行列の行内の値が全て0となる場合への対応

```
7×7 Matrix{Float64}:
 -0.975585   -0.789227   -0.296749        0.31374        0.809054       1.0              0.813643
  0.0        -0.337116   -0.544575       -0.543134      -0.333343       0.00466387       0.34178
  0.0         0.0         1.95713e-5      5.12382e-5     8.29052e-5     0.000102476      0.000102476
  0.0         0.0         0.0             1.0            0.0            0.0              0.0
  0.0         0.0         0.0             0.0            1.52344e-15    2.42994e-15      2.73085e-15
  0.0         0.0         0.0             0.0            0.0            1.0              0.0
  0.0         0.0        -3.38813e-21     0.0            0.0            0.0             -9.40416e-16
julia> B'
1×7 adjoint(::Vector{Float64}) with eltype Float64:
 -0.784639   0.338007   1.95713e-5   0.0   1.1226e-15   0.0   -9.15407e-16
julia> #-------------------------------- sweepout backward
julia> for i in M:-1:2
            if abs(A[i,i]) < myeps
                A[i,:]=zeros(M)
                B[i]=0
                A[i,i]=1
            end
            for j=1:i-1
                mx=A[j,i]/A[i,i]
                A[j,:] .-= mx*A[i,:]
                B[j] -= mx*B[i]
            end
            B[i] /= A[i,i]
            A[i,:] /= A[i,i]
        end
julia>   B[1] /= A[1,1];
julia>   A[1,:] /= A[1,1];
julia> A
7×7 Matrix{Float64}:
  1.0   -0.0   -0.0   -0.0   -0.0   -0.0   -0.0
 -0.0    1.0   -0.0   -0.0   -0.0   -0.0   -0.0
  0.0    0.0    1.0    0.0    0.0    0.0    0.0
  0.0    0.0    0.0    1.0    0.0    0.0    0.0
  0.0    0.0    0.0    0.0    1.0    0.0    0.0
  0.0    0.0    0.0    0.0    0.0    1.0    0.0
  0.0    0.0    0.0    0.0    0.0    0.0    1.0
julia> B'
1×7 adjoint(::Vector{Float64}) with eltype Float64:
 0.688853   1.69575   -2.12148   0.0   0.736882   0.0   0.0
julia> #------------------------ remove null higher orders a_m (m>M′ )
julia> M′  =M;
julia> for i in 1:M
            if abs(B[i]) > myeps
                M′  =i
```

丸め誤差により0になれていない

続くステップで
ギリギリ0判定

この行は0判定されるべきものであるが、myepsが小さすぎるために判定漏れとなっている。
これにより無駄にランクが2つ上がる。

ここの数値は本来全て0となるべきもの

行列の行内の値が全て0となる場合への対応。
上で掃き出しきれなかった最終行がここで
クリアされ、余分なランクが2つ下げられる。

正しくランクが下がっている部分

myepsが小さすぎたためにランクが下がり切れていない部分。
これらは本来0になるべきもの。

正弦波と定数項に対応する項。
これら3つのみが残るのが正しい。

```
            end
        end
julia> M′
5
julia> #------------------------------ set prediction coeffs. & get modes
julia> predcoeffs=ones(M′ +1);
julia> for i in 1:M′
            predcoeffs[i]=-B[M′ +1-i]
        end
julia> predcoeffs'
1×6 adjoint(::Vector{Float64}) with eltype Float64:
 -0.736882   -0.0   2.12148   -1.69575   -0.688853   1.0
julia> Polynomial(predcoeffs)
Polynomial(-0.7368817567792972 + 2.121481213065832*x^2 - 1.6957469524556559*x^3
                - 0.6888525038427817*x^4 + 1.0*x^5)
julia> modes=roots(Polynomial(predcoeffs))
5-element Vector{ComplexF64}:
 -1.4045379479974356 + 0.0im
 -0.5246435369181329 + 0.0im
  0.8090169943749257 - 0.5877852522924136im
  0.8090169943749257 + 0.5877852522924136im
    1.000000000008501 + 0.0im
julia> #------------ remove exterme modes which corrrespond to noise floor
julia> MaxDiffBetweenEdges=100;
julia> M′ ′ =0;
julia> for i in 1:M′
            growwidth = abs(modes[i])^TotN
            if maximum([growwidth, 1/growwidth]) < MaxDiffBetweenEdges
                M′ ′ +=1
                modes[M′ ′ ]=modes[i]
            end
        end
julia> modes=modes[1:M′ ′ ]
4-element Vector{ComplexF64}:
 -1.4045379479974356 + 0.0im
  0.8090169943749257 - 0.5877852522924136im
  0.8090169943749257 + 0.5877852522924136im
    1.000000000008501 + 0.0im
julia> M′ ′
4
julia> #---------- calc complex amps. at left bound solve AX=B
julia> A=zeros(ComplexF64, M′ ′ ,M′ ′ );
julia> B=zeros(ComplexF64, M′ ′ );
julia> for i in 1:M′ ′ , j in 1:M′ ′ , n in 0:TotN-1
            A[i,j]    += (modes[i]*modes[j])^n
```

本来残ってほしい予測係数はこの部分

ここは本当は3次関数になって欲しい

解析次数が下がり切れなかったために残った項。区間両端での振幅差が極端に大きい項となる場合が多い。この特性を利用して次のプロセスで除去するが、その閾値(MaxDiffBetweenEdges)が大きすぎるために除去しきれない状況を以下で例示している。

正弦波と定数項に対応する項

不適切な閾値設定

区間両端での振幅差が極端に大きくなる項を除去している。

上記閾値判定ですり抜けた項

閾値判定でのすり抜けがあるものの、ランクがさらに1つ下がってM''=4

振幅計算は複素行列の対角化になる。

```
        end
julia> A
4×4 Matrix{ComplexF64}:
   3570.16+0.0im        -15.8329+20.1204im    -15.8329-20.1204im    -24.0956+0.0im
 -15.8329+20.1204im    1.30902-0.951057im       12.0+0.0im          1.80902-0.587785im
 -15.8329-20.1204im       12.0+0.0im         1.30902+0.951057im    1.80902+0.587785im
 -24.0956+0.0im         1.80902-0.587785im    1.80902+0.587785im       12.0+0.0im
julia> for i in 1:M′ ′ , n in 0:TotN-1
            B[i] += ts[n+1]*modes[i]^n
        end
julia> B
4-element Vector{ComplexF64}:
 -20.361325254376972 + 0.0im
 0.49361842809224077 - 5.351369355333874im
 0.49361842809224077 + 5.351369355333874im
  0.7077852522222327 + 0.0im
julia> iCAmp = A ¥ B
4-element Vector{ComplexF64}:
 1.4140565131219918e-15 + 2.487784153979577e-19im
 7.051881123245964e-14 + 0.5000000000002429im
 7.052191153141293e-14 - 0.5000000000002429im
  0.009999999999492205 - 6.100912414782267e-21im
julia> #------------------   prepare return values
julia> iFrq = imag(log.(modes))/2π;
julia> iAVR = real(log.(modes));
julia> results=[];
julia> for i in 1:M′ ′
            push!(results,(iFrq[i],iAVR[i],iCAmp[i]))
        end
julia> results
4-element Vector{Any}:
 (0.5, 0.339708386063257, 1.4140565131219918e-15 + 2.487784153979577e-19im)
 (-0.09999999999999439, -5.2569060216003926e-14, 7.051881123245964e-14 + 0.5000000000002429im)
 (0.09999999999999439, -5.2569060216003926e-14, 7.052191153141293e-14 - 0.5000000000002429im)
 (0.0, 8.50097769951869e-12, 0.009999999999492205 - 6.100912414782267e-21im)
julia> #------------------ plot spectrum of each mode
julia> """ equation for lorenz profile spectrum """;
julia> LorenzProf(f,iFrq,iAVR,iCAmp)=
            abs(iCAmp)*√(iAVR^2/((2π*(abs(iFrq)-f))^2+iAVR^2));
julia> x=0:0.00001:0.2;
julia> figure();
julia> for i in 1:M′ ′
        #   y=LorenzProf.(x,iFrq[i],iAVR[i],iCAmp[i]);
            y=LorenzProf.(x,results[i][1],results[i][2],results[i][3]);
            plot(x,y,label="mode no. $i");
```

連立方程式の解の計算

閾値判定ですり抜けた項の振幅。
非常に小さい値になるので、後処理で除去する。
（この例示内では除去しない）

正弦波と定数項に対応する項の振幅

順に、瞬時周波数、瞬時振幅増減率、瞬時複素振幅

スペクトル関数の定義式

どちらの行をとっても同じプロットになる。
表現の違いのみ。

```
        end
julia> yscale("log"); legend(); show(); #gcf()
```