Generalized Factor Analysis for Time Series

Bill Menke, October 26, 2022

Statement of problem. Suppose that there are:

N_z data time series, $\mathbf{z}^{(i)}$, each of length N_t;
N_v factor time series, $\mathbf{v}^{(i)}$, each of unit variance and length N_t, with $N_v \ll N_z$; and
$N_v \times N_f$ filters, $\mathbf{f}^{(i,j)}$, each of length N_f, with $N_f \ll N_t$

We seek to represent data time series, $\mathbf{z}^{(i)}$, as a sum of filtered versions of $\mathbf{v}^{(j)}$:

$$\mathbf{z}^{(i)} \approx \sum_{j=0}^{N_v-1} \mathbf{f}^{(i,j)} * \mathbf{v}^{(j)} \tag{1}$$

Here $*$ denotes convolution.

Non-uniqueness of Solution. Note that this representation is non-unique, because we can insert any all-pass filter, $\mathbf{g}^{(j)}$ and any $N_v \times N_v$ unary matrix, $\mathbf{M}$,without changing $\mathbf{z}^{(i)}$. That is,

$$\mathbf{z}^{(i)} \approx \sum_{j=0}^{P-1} \left[\left(\mathbf{f}^{(i,j)} * \left[\mathbf{g}^{(j)}\right]^{inv} \right) \mathbf{M}^T \right] * \left[\mathbf{M}\left(\mathbf{g}^{(j)} * \mathbf{v}^{(j)} \right) \right] \tag{2}$$

The conditions on $\mathbf{g}$ and $\mathbf{M}$ are needed to preserve the variance of $\mathbf{v}^{(j)}$.

Solution Method. The following version of Newton's method estimates $\mathbf{f}^{(i,j)}$ and $\mathbf{z}^{(i)}$ (but does not necessarily converge to an optimum solution):

Step 1: Define the matrix $Z_{ij} = z_j^{(i)}$ and take compute the singular value decomposition,

$$\mathbf{Z} = \mathbf{US}\left[\mathbf{v}^{(0)} \quad \mathbf{v}^{(1)} \quad \dots \quad \mathbf{v}^{(N_v-1)} \quad \dots \quad \mathbf{v}^{(N_t-1)}\right]^T \tag{3}$$

Select the N_v **v**s with the largest singular values, S_{ii}, and normalize them to unit variance.

Step 2: Holding $\mathbf{v}^{(j)}$ fixed, solve (1) for $\mathbf{f}^{(i,j)}$ via least squares.

Step 3: Holding $\mathbf{f}^{(i,j)}$ fixed, and solve (1) for $\mathbf{v}^{(j)}$ via least squares.

Step 4: Normalize the **v**s to unit variance.

Then iterate over steps 2-4 until the error of (1) stops decreasing.

Example. We set $N_t = 128$, $N_z = 5$, $N_v = 3$ and $N_f = 16$, construct true **v**s and **f**s using random number generators, and compute true **z**s by evaluating (1). We then use the method to estimate the **v**s and **f**s from the **z**s.

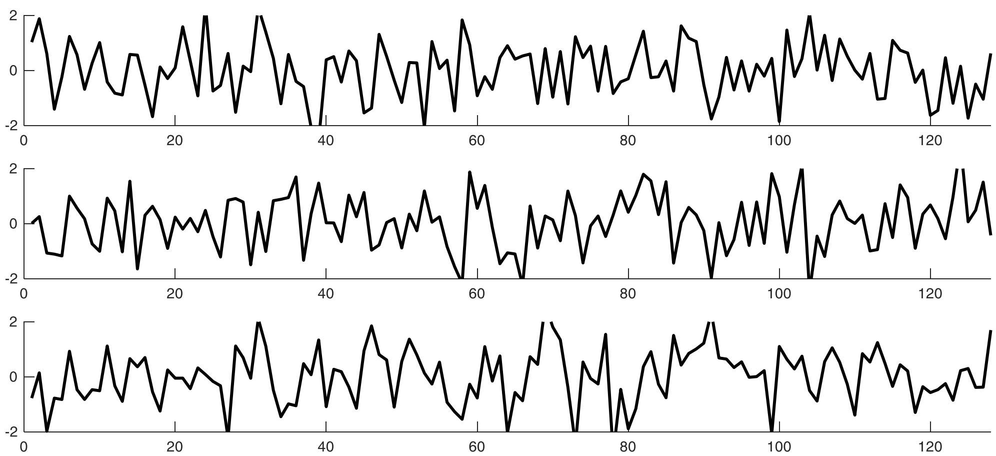

Figure 1. True factor timeseries are Normally-distributed random numbers.

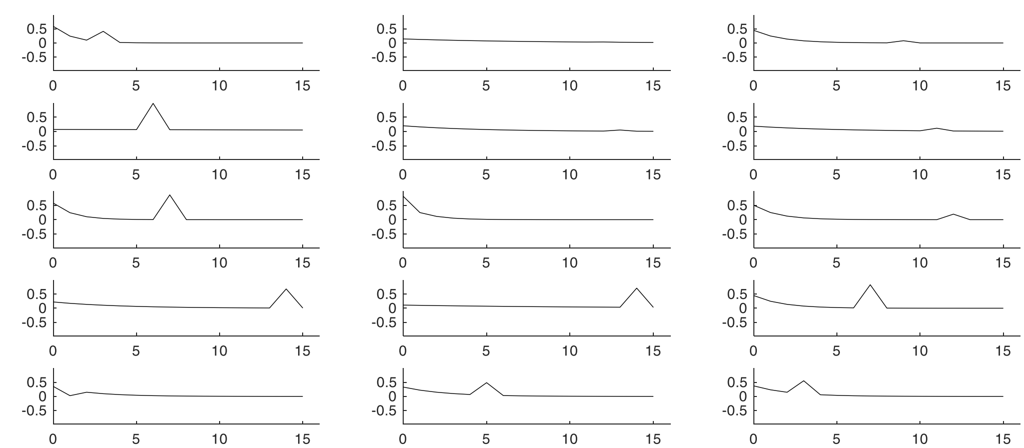

Figure 2. True filters are the sum of a pulse with randomly-chosen amplitude at a randomly-chosen position and exponential decay with a random-chosen decay rate.

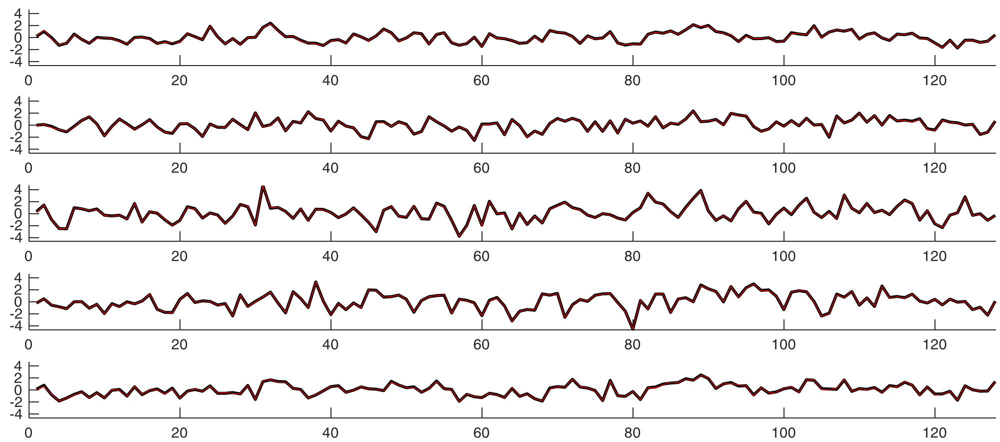

Figure 3. True data time series, $\mathbf{z}^{(i)}$ (black) and their estimated versions (red).

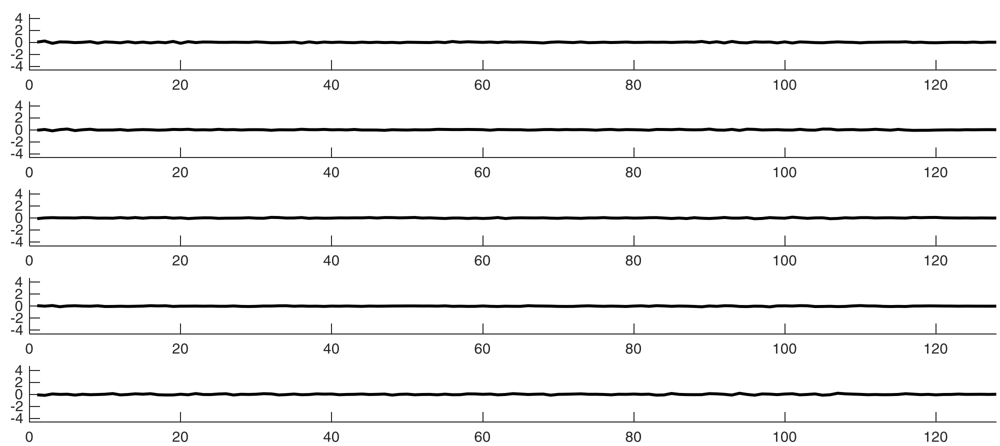

Figure 4. Data time series prediction error.

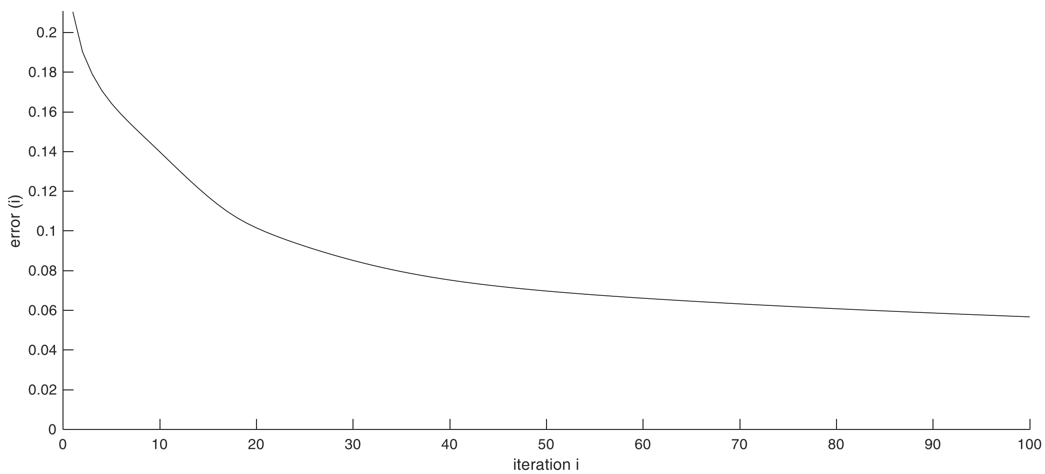

Figure 5. Root mean squared error of Equation 1 as a function of iteration number, i.

An ensemble of tests, one of which is shown Figures 1-4, indicates that the method generally converges to a reasonably good solution, but that it rarely converges to an exact solution. An examination of prediction error indicates that the lower frequencies are most poorly fit.

Test Code.

```
% Generalized factor analysis for time series
% Bill Menke, October 26, 2022
clear all;

Nz = 5;
Nt = 128;
Nv = 3;
Nf = 16;

Niter=100;

% build test factors, normalized to unit variance
v = random('Normal',0,1,Nt,Nv);
for i=[1:Nv]
    v(:,i) = v(:,i)/std(v(:,i));
end

% build test filters
if(1) % exponetial decay on 1, random on 0
smax = 1;
s = random('uniform',0,smax,Nz,Nv);
f = zeros(Nf,Nz,Nv);
for i=[1:Nz]
for j=[1:Nv]
    f(:,i,j) = exp( -s(i,j)*[0:Nf-1]' );
    fsum = sum(f(:,i,j));
    f(:,i,j) = f(:,i,j)/fsum;
    k = randi([1,Nf]);
```

```
    f(k,i,j) = random('uniform',0,1);

end
end
else
    f= random('Uniform',-1,1,Nf,Nz,Nv);
end
fabsmax = max(abs(f(:)));

figure(5);
clf;
for i=[1:Nz]
for j=[1:Nv]
    subplot(Nz,Nv,(i-1)*Nv+j);
    hold on;
    axis([0,Nf,-fabsmax,fabsmax]);
    plot([0:Nf-1],f(:,i,j),'k-');
end
end

% construct test data timeseries
z = zeros(Nt,Nz);
for i=[1:Nz]
    for j=[1:Nv]
        tmp = conv(v(:,j),f(:,i,j));
        z(1:Nt,i) = z(1:Nt,i) + tmp(1:Nt);
    end
end

zabsmax = max(abs(z(:)));

% step 1: initialize factor timeseries to right
% eigenvectors of SVD of z-matrix, normalized to
% unit variance
[U, L, vp] = svd(z',0);
vp = vp(:,1:Nv);
for i=[1:Nv]
    vp(:,i) = vp(:,i)/std(vp(:,i));
end

% iterations

E = zeros(Niter,1);
for k=[1:Niter]

% step 2, solve for filters, holding factor timeseries fixed
% zp = vp(const) * fp(unknown);
fp = zeros(Nf,Nz,Nv);
for i=[1:Nz]
    G = [];
    for j=[1:Nv]
        c = vp(:,j);
        r = [vp(1,j),zeros(1,Nf-1)];
```

```
        M = toeplitz( c, r );
        G = [G,M];
    end
    m = (G'*G)\(G'*z(:,i));
    for j=[1:Nv]
        le = 1+(j-1)*Nf;
        rt = Nf+(j-1)*Nf;
        fp(1:Nf,i,j) = m(le:rt,1);
    end
end

% step 3, colve for factor tineseries, holding filters fixed
% zp = fp(const) * vp(unknown)
% [zp1] = [fp11 fp12] [vp1]
% [zp2] = [fp21 fp22] [vp2]
% [zp3] = [fp31 fp32]
% [zp4] = [fp41 fp42]
d = [];
for i=[1:Nz]
    d = [d; z(:,i) ];
end
G = [];
for i=[1:Nz]
    GR = [];
    for j=[1:Nv]
        c = [fp(:,i,j); zeros(Nt-Nf,1)];
        r = [fp(1,i,j),zeros(1,Nt-1)];
        M = toeplitz( c, r );
        GR = [GR,M];
    end
    G = [G;GR];
end
m = (G'*G)\(G'*d);
for i=[1:Nv]
    le = 1+(i-1)*Nt;
    rt = Nt+(i-1)*Nt;
    vp(:,i) = m(le:rt);
    vp(:,i) = vp(:,i)/std(v(:,i));
end

% reconstruct data time series
zp = zeros(Nt,Nz);
for i=[1:Nz]
    for j=[1:Nv]
        tmp = conv(vp(:,j),fp(:,i,j));
        zp(1:Nt,i) = zp(1:Nt,i) + tmp(1:Nt);
    end
end

% calculate error
for i=[1:Nz]
      e = z(:,i)-zp(:,i);
      E(k) = E(k) + e'*e;
```

```
end

end % end of iterations

% plot z (black) and zp (red)
figure(1);
clf;
for i=[1:Nz]
    subplot(Nz,1,i);
    hold on;
    axis([0,128,-zabsmax,zabsmax]);
    plot([1:Nt],zp(:,i),'k-','LineWidth',2);
    plot([1:Nt],z(:,i),'r-');
end

% plot error
figure(2);
clf;
for i=[1:Nz]
    subplot(Nz,1,i);
    hold on;
    axis([0,128,-zabsmax,zabsmax]);
    plot([1:Nt],z(:,i)-zp(:,i),'k-','LineWidth',2);
end

% plot total error
figure(3);
clf;
hold on;
sigmaz = sqrt(E/(Nt*Nz));
axis( [0, Niter, 0, max(sigmaz)] );
plot([1:Niter]',sigmaz,'k-');
xlabel('iteration i');
ylabel('error (i)');

% plot factor timeseries
if(0)
figure(4);
clf;
for i=[1:Nv]
    subplot(Nv,1,i);
    hold on;
    axis([0,128,-2,2]);
    plot([1:Nt],v(:,i),'k-','LineWidth',2);
end
end
```