Wiener Filtering by CKMS+ Development (displays)

October 20, 2021

1 Preformance of CKMS+

I keep all my scripts out of the way in a file in this directory.

[23]:

```
using DSP, PyPlot, Polynomials, FFTW, Statistics

sc = include("WFbyCKMSplus_scripts.jl")
at = include("../AnalysisToolbox.jl")
mrb = include("../WFMR_bs.jl")
mr  = include("../WFMR.jl")

whf = include("../WhiteningFilters.jl")
util = include("../Utils.jl")
```

```
WARNING: replacing module WFbyCKMSplus_scripts.
WARNING: replacing module AnalysisToolbox.
WARNING: replacing module WFMR_bs.
WARNING: replacing module WFMR.
WARNING: replacing module WhiteningFiltersScalar.
WARNING: replacing module VariousUtilities.
```

[23]:

```
Main.VariousUtilities
```

As a working test case I pick the AR(2) process $y_n = 5/4y_{n-1} - 3/8y_{n-2} + e_n$ as signal to be estimated and $x_n = y_n + u_n$

[ ]:

```
N = 10^4; D = 10^3; p = 2;
e = randn(N+D)

Zeros = 2*rand(10) .- 1
Poles = 2*rand(10) .- 1
```

[36]:

```
[Zeros Poles]
```

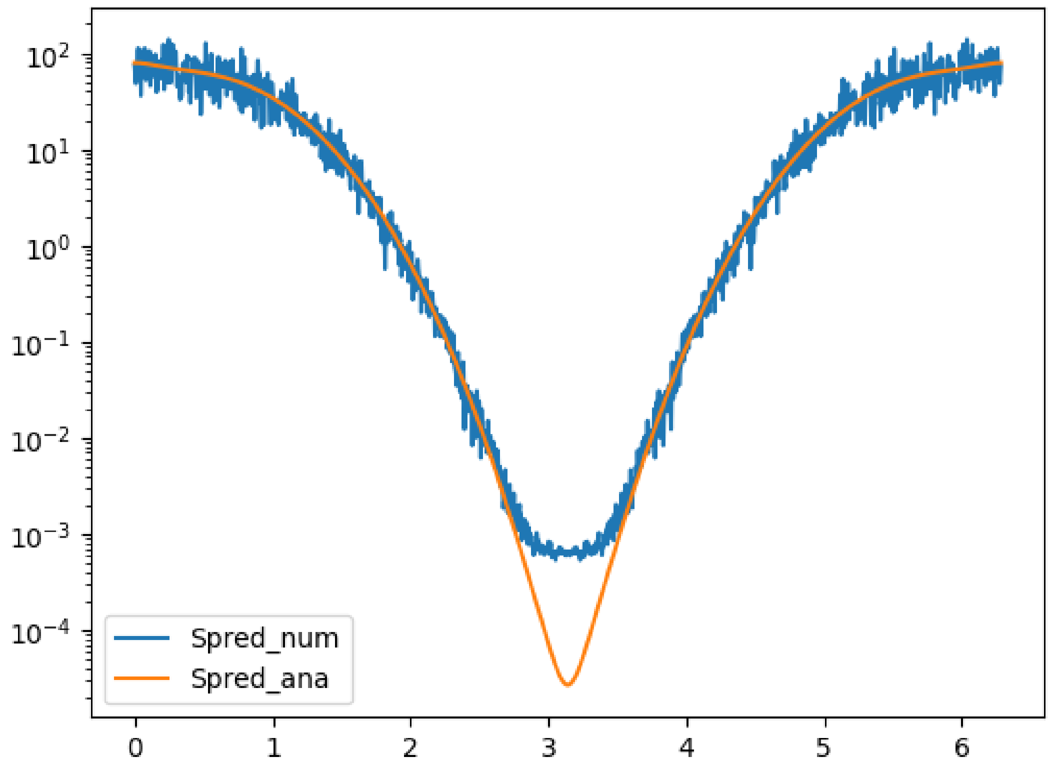
[36]:

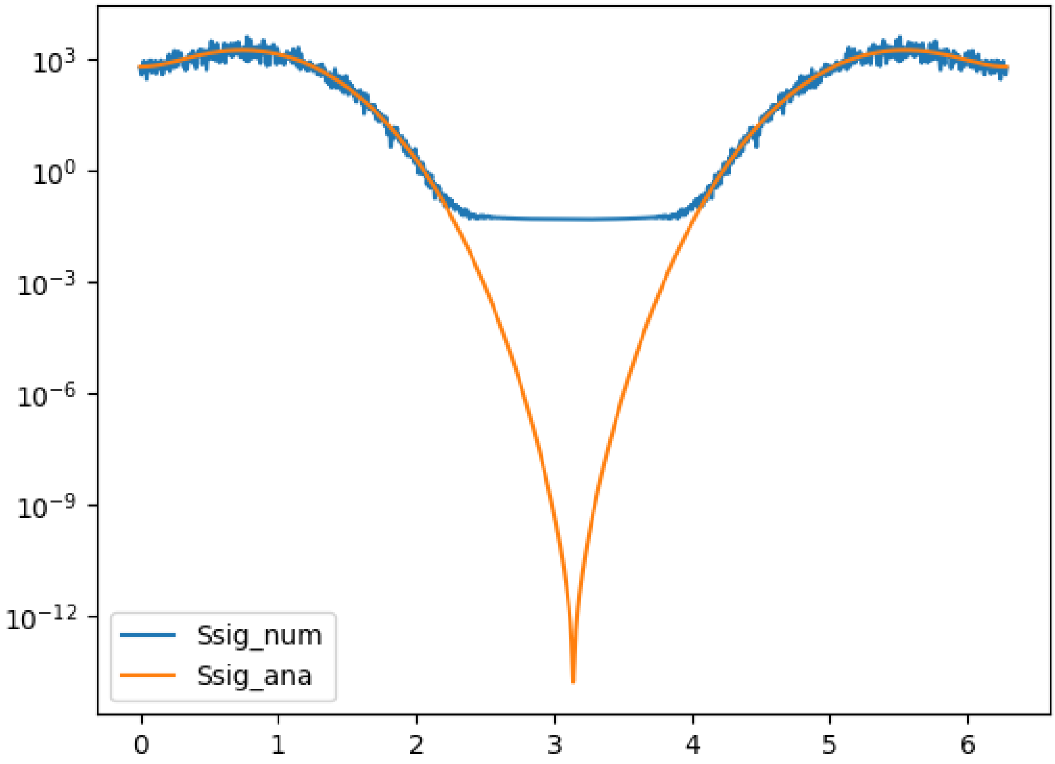
```
10×2 Array{Float64,2}:
  0.492145   -0.684361
 -0.644545   -0.572419
 -0.768893   -0.0984303
```

```
  0.928019    0.326703
 -0.130403   -0.221751
 -0.0837919  -0.406385
  0.715443   -0.467725
  0.0174513   0.612506
 -0.615667   -0.874505
 -0.755121    0.690267
```

[33]:

```
f = coeffs(Polynomial([1])*prod(Polynomial([1, -z]) for z in Zeros))
w = coeffs(Polynomial([1])*prod(Polynomial([1, -z]) for z in Poles))
r = 1.0

pred = at.ARMA_gen(;l, w, r, e, steps = N, discard = D);

A = 2*rand(10) .- 1
f = coeffs(Polynomial([1])*prod(Polynomial([1, -a]) for a in A))
sig = filt(f,pred)

Nex = 2^13; Θ = 2pi*(0:Nex-1)/Nex;
Spred_num = at.z_crossspect_dm(pred,pred;Nex, L = 1500)
Hpred(z) = Polynomial(w)(z)/Polynomial(l)(z)
Spred_ana = map(z -> abs2(Hpred(z)),exp.(im*Θ))

semilogy(Θ,Spred_num, label = "Spred_num")
semilogy(Θ,Spred_ana, label = "Spred_ana")
legend()

Ssig_num = at.z_crossspect_dm(sig,sig;Nex)
Hsig(z) = Polynomial(f)(z)*Hpred(z)
Ssig_ana = map(z -> abs2(Hsig(z)),exp.(im*Θ))

figure()
semilogy(Θ,Ssig_num, label = "Ssig_num")
semilogy(Θ,Ssig_ana, label = "Ssig_ana")
legend()

sig  = reshape(sig, 1,:)
pred = reshape(pred, 1,:)

[sig; pred]
```


10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
1
2
3
4
5
6
Spred_num
Spred_ana


10^{3}
10^{0}
10^{-3}
10^{-6}
10^{-9}
10^{-12}
0
1
2
3
4
5
6
Ssig_num
Ssig_ana

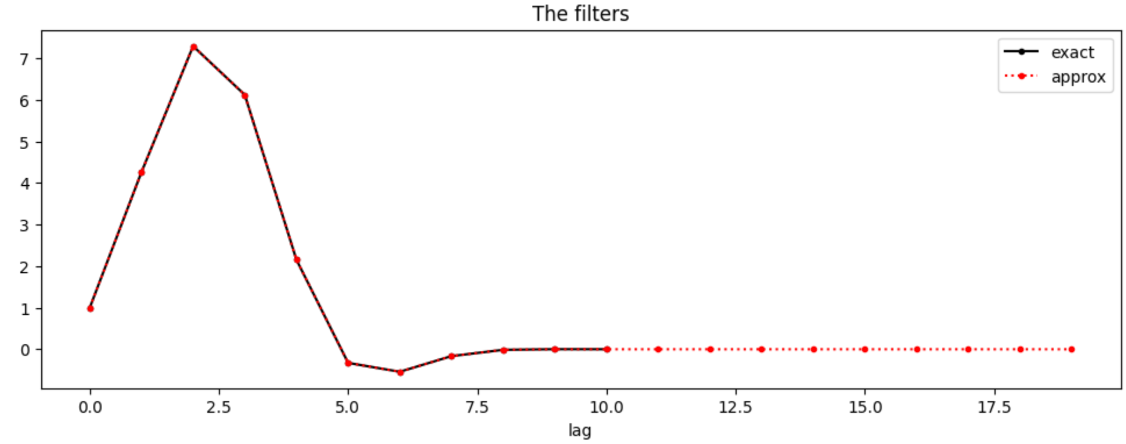
[33]:
```
2×10000 Array{Complex{Float64},2}:
 2.25355+0.0im  7.46571+0.0im  …   -41.135+0.0im  -49.0804+0.0im
 2.25355+0.0im  1.49061+0.0im     -9.71788+0.0im  -7.18813+0.0im
```

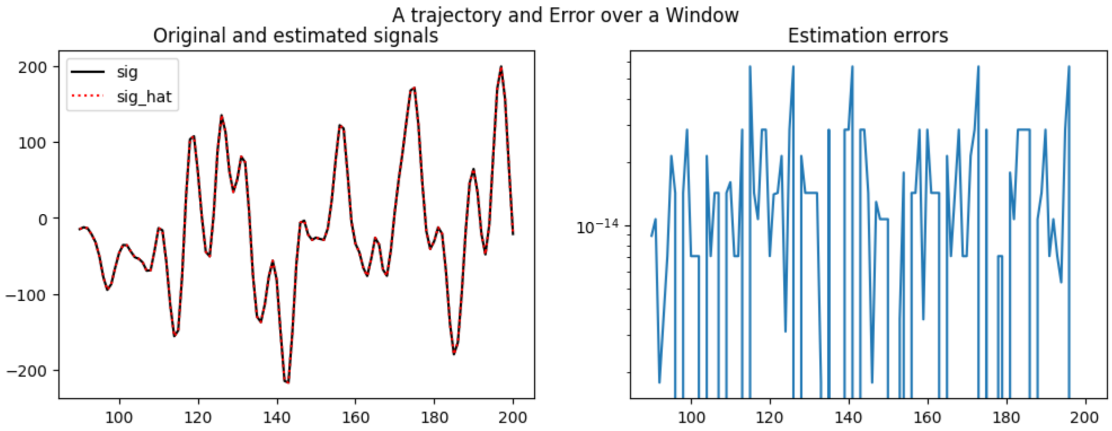
[28]:
```
function testdisp(h,sig,pred; vew = 90:200, f)
    figure(figsize=(12,4))
    title("The filters")
    f == 0 || plot(f,"k.-",label = "exact")
    plot(h[:],"r.:",label = "approx")
    xlabel("lag"); legend()

    sig_hat = at.my_filt(h,pred);

    vew = 90:200
    res = util.TakeLook(sig,sig_hat; vew)
    suptitle("A trajectory and Error over a Window")
    println("MSE: ",var(res[100:end]))
end
```

[28]:
```
testdisp (generic function with 1 method)
```

1.1 Backslash

[29]:
```
# Benchmark
h = @timed mrb.get_wf_bs(sig,pred; M_out = 20)
tim = h.time; h = h.value

testdisp(h,sig,pred; f)

println("Time: ",tim," sec")
```



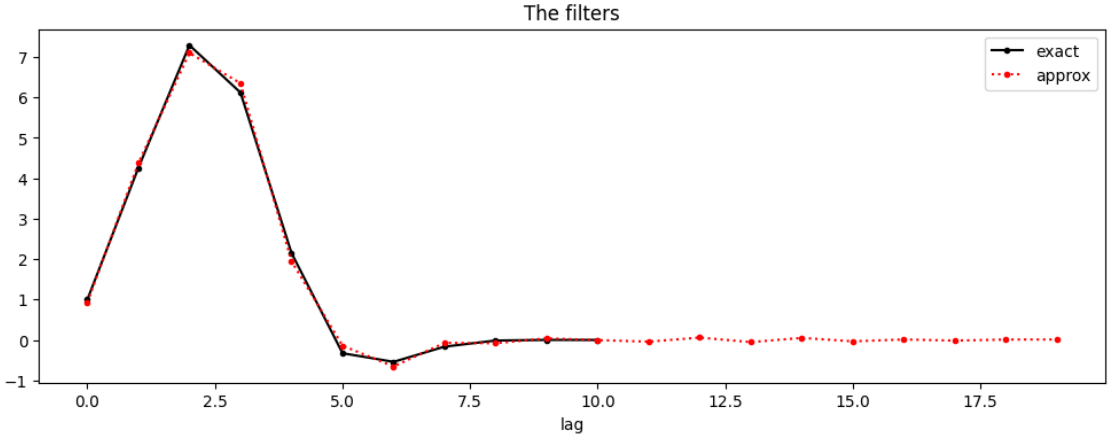


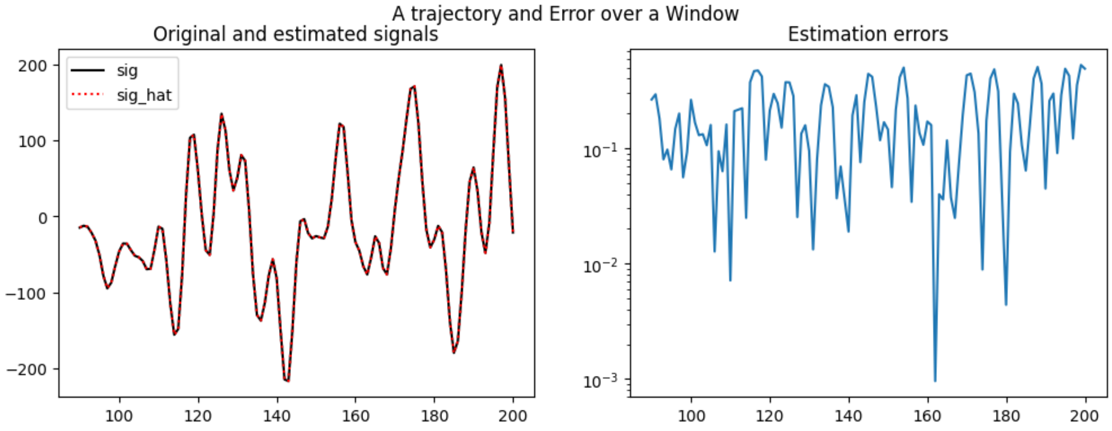
```
MSE: 1.9149153463491939e-28
Time: 0.097540572 sec
```

1.2 Old CKMS

[30]:

```
# Benchmark
h = @timed mr.get_wf(sig,pred; M_out = 20)
tim = h.time; h = h.value

testdisp(h,sig,pred; f)

println("Time: ",tim," sec")
```





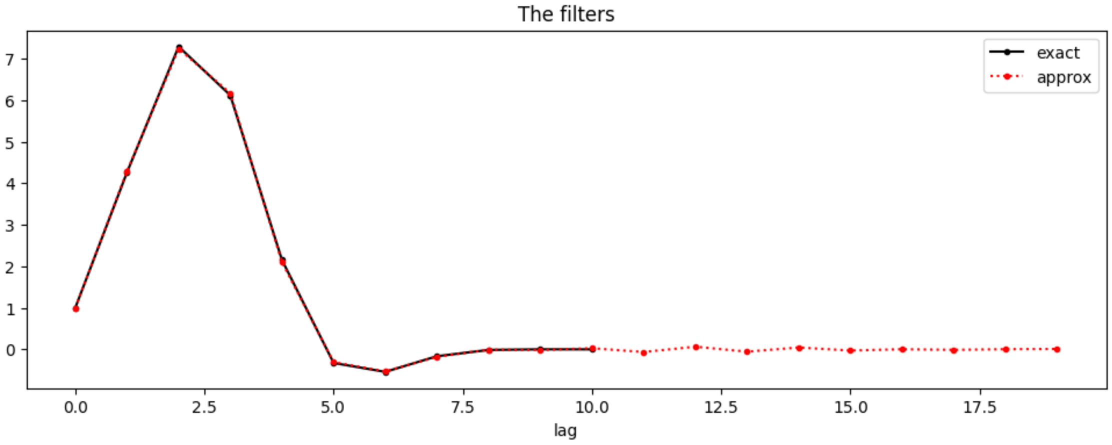
```
MSE: 0.022002109838818 98
Time: 1.02694499 sec
```

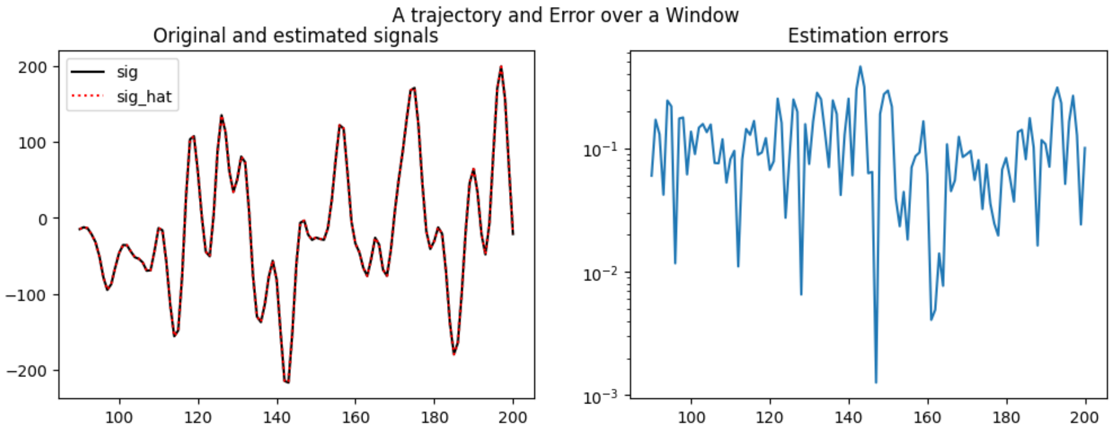
1.3 CKMS+ (1 iteration)

[31]:

```
# Benchmark
h = @timed sc.vector_wiener_filter_fft(sig,pred, maxit = 1)
tim = h.time; h = h.value

testdisp(h,sig,pred; f)

println("Time: ",tim," sec")
```

```
Time taken for crossspect: 2.430972521
Bytes Allocated: 8750418544
Time taken for spectfact: 0.111565376
Bytes Allocated: 181525392
```





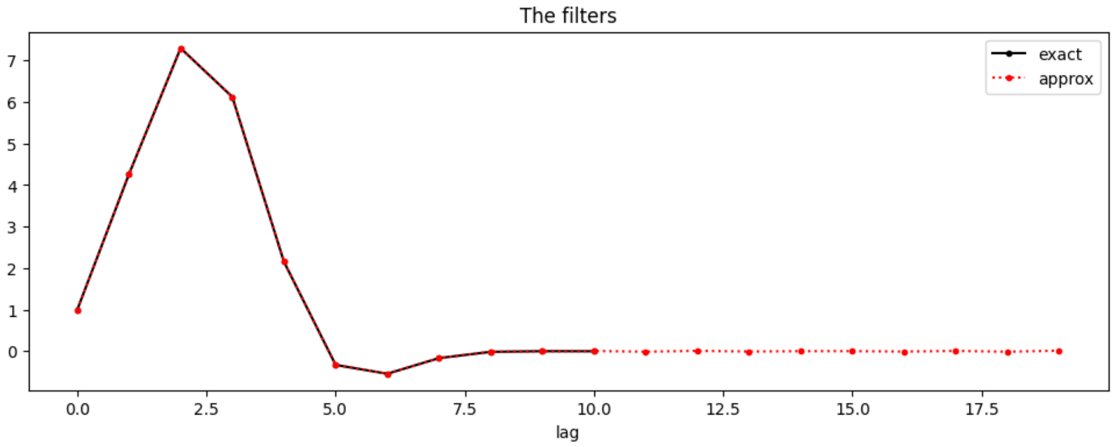
```
MSE: 0.006568257117226377
Time: 2.621411856 sec
```

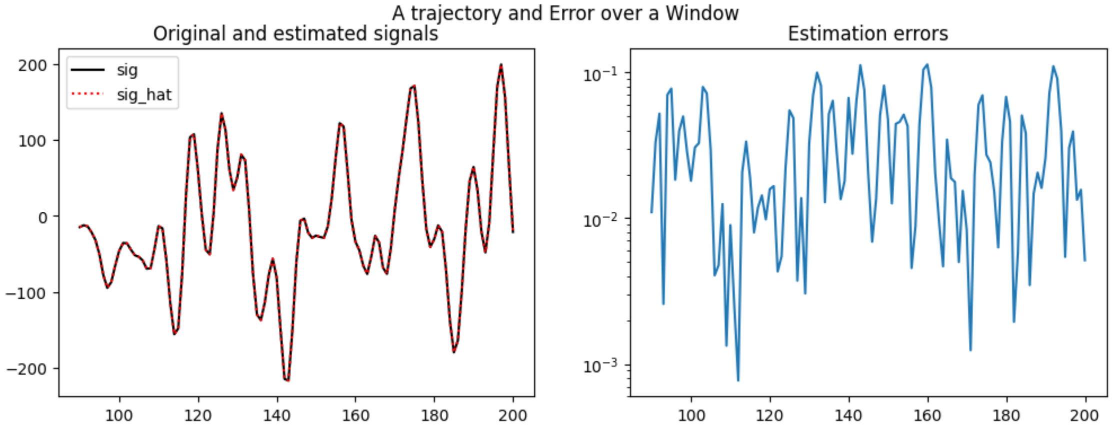
1.4 CKMS+ (2 iterations)

[32]:

```
# Benchmark
h = @timed sc.vector_wiener_filter_fft(sig,pred, maxit = 2)
tim = h.time; h = h.value

testdisp(h,sig,pred; f)

println("Time: ",tim," sec")
```

```
Time taken for crossspect: 8.218663474
Bytes Allocated: 23151236128
Time taken for spectfact: 0.173399742
Bytes Allocated: 313831968
```





```
MSE: 0.0006449172240871781
Time: 8.47810017 sec
```

[ ]: