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")
```

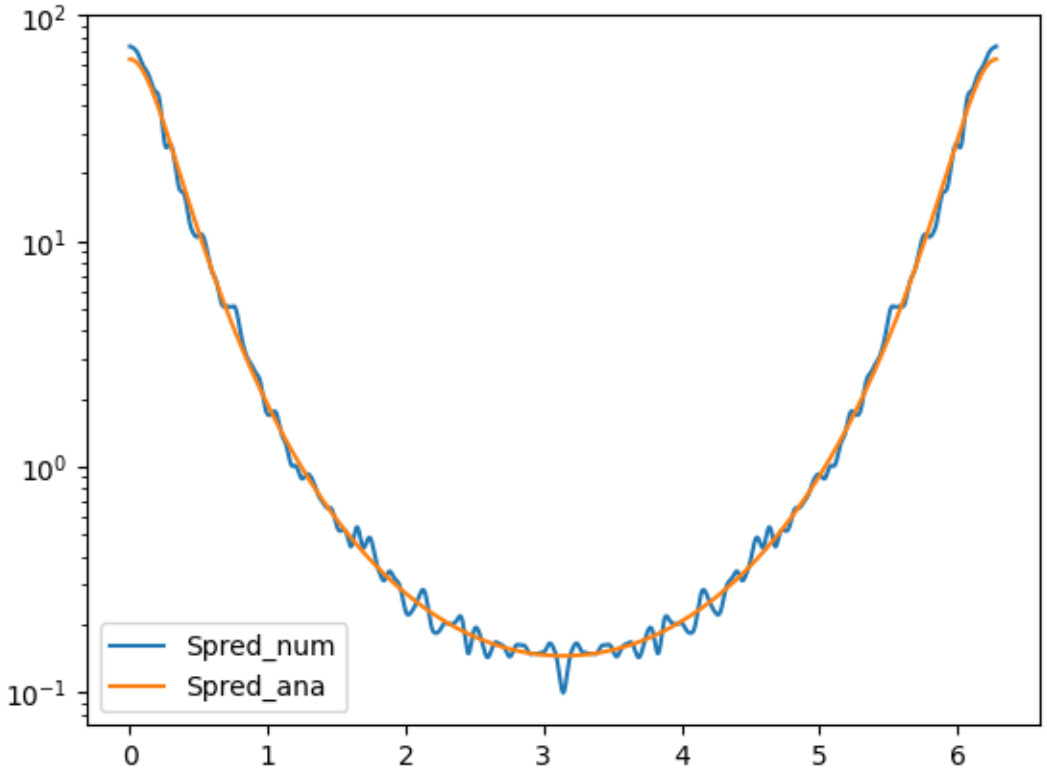
```
WARNING: replacing module WFbyCKMSplus_scripts.
WARNING: replacing module AnalysisToolbox.
WARNING: replacing module WFMR_bs.
WARNING: replacing module WFMR.
WARNING: replacing module WhiteningFiltersScalar.
WARNING: replacing module VariousUtilities.
```

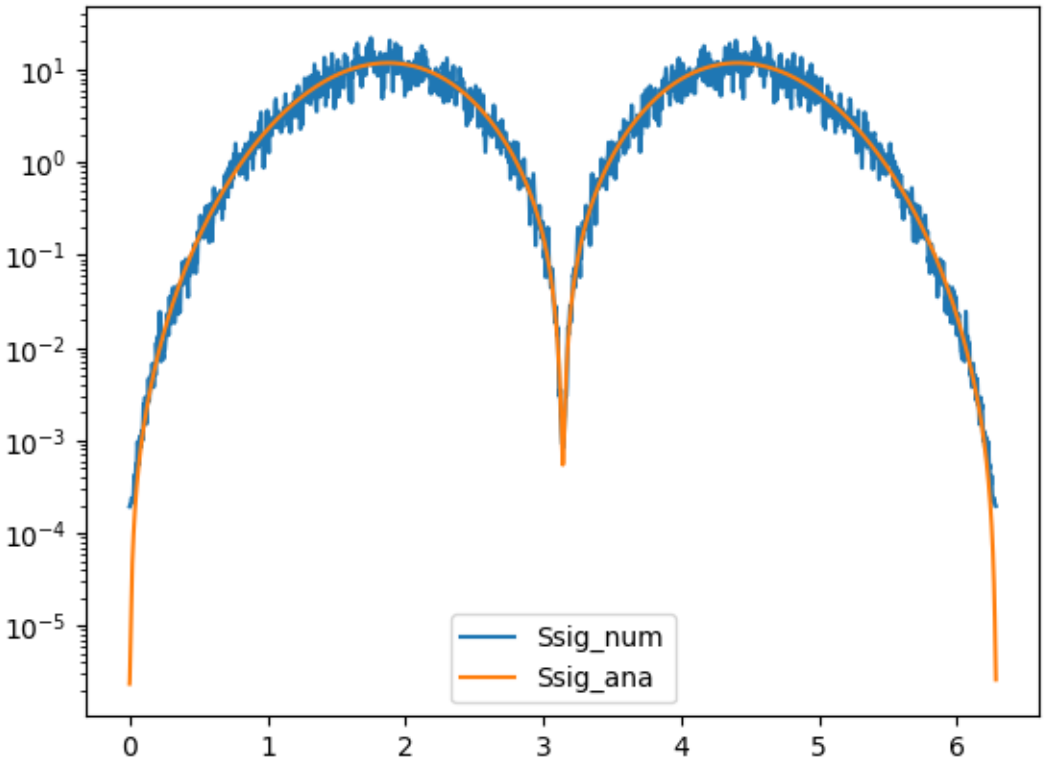
[23]:
```
Main.VariousUtilities
```

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 $x_n = y_n + u_n$

[16]:
```
N = 10^4; D = 10^3; p = 2;
e = randn(N+D)

l = [1, -5/4, 3/8]
w = [1]
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 = 2^12; θ = 2pi*(0:Nex-1)/Nex;
Spred_num = at.z_crossspect_dm(pred,pred;Nex, L = 150)
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]
```





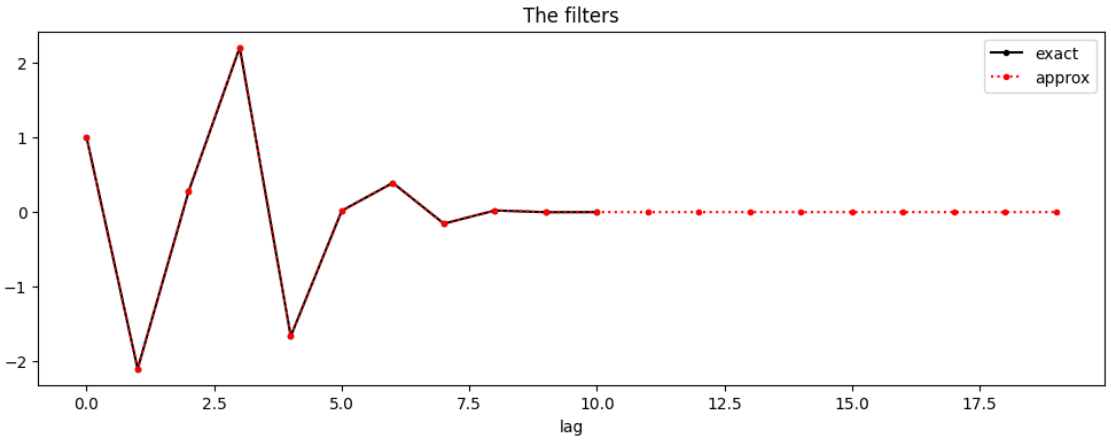
[16]:
```
2×10000 Array{Complex{Float64},2}:
 -2.41637+0.0im   1.14022+0.0im  …  -2.56645+0.0im   3.16904+0.0im
 -2.41637+0.0im  -3.93579+0.0im     -2.98996+0.0im  -1.36057+0.0im
```

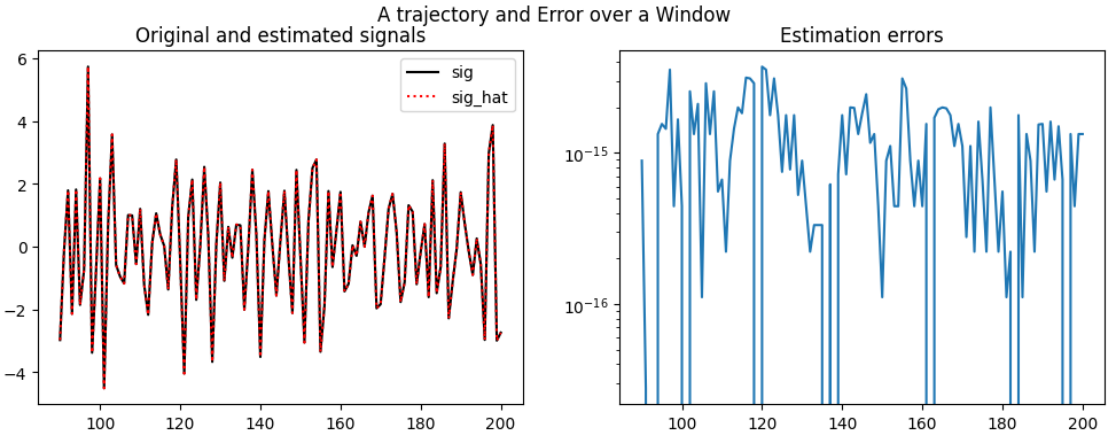
[17]:
```
function testdisp(h,sig,pred; vew = 90:200, f)
    figure(figsize=(12,4))
    title("The filters")
    f == 0 || plot(f,"k.-",label = "exact")
    plot(h[:],"r.:",label = "approx")
    xlabel("lag"); legend()

    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
```

[17]:
```
testdisp (generic function with 1 method)
```

1.1 Backslash

[18]:
```
# 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")
```

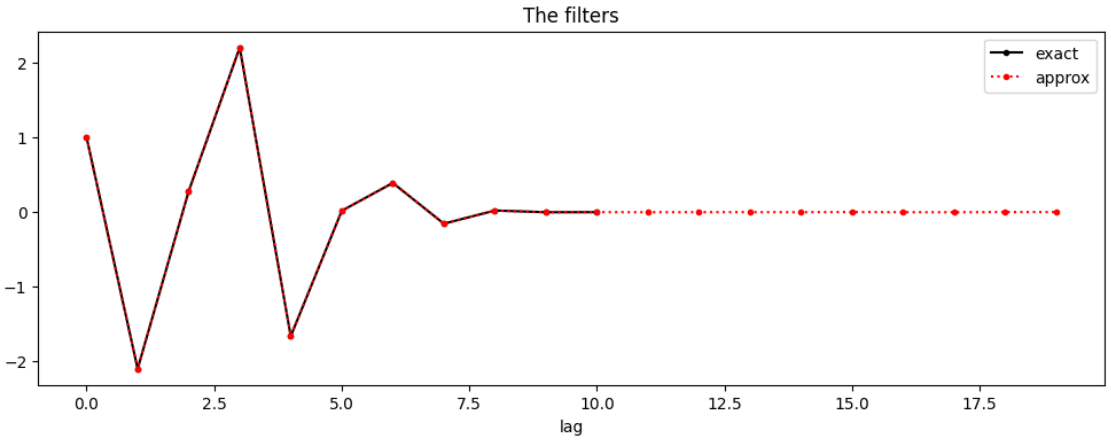


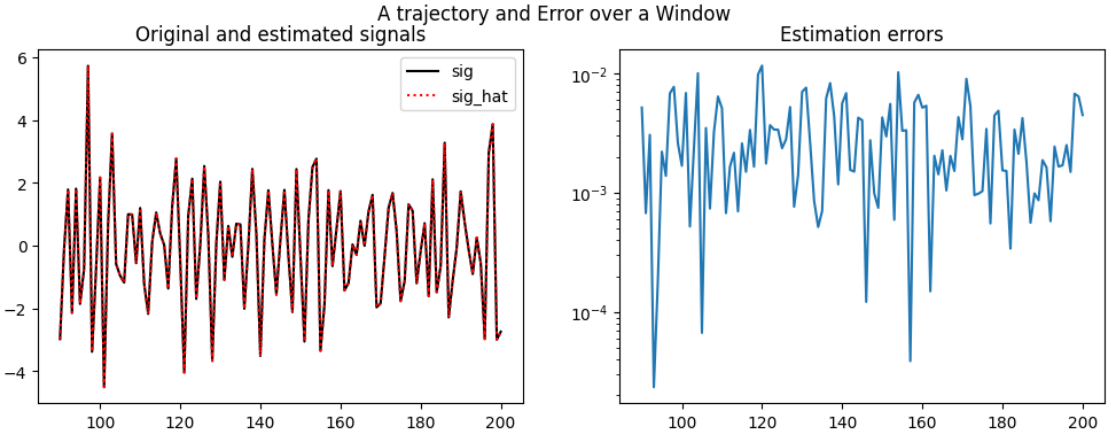



```
MSE: 1.0267528585912878e-30
Time: 0.096035977 sec
```

1.2 Old CKMS

[19]:
```
# 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")
```



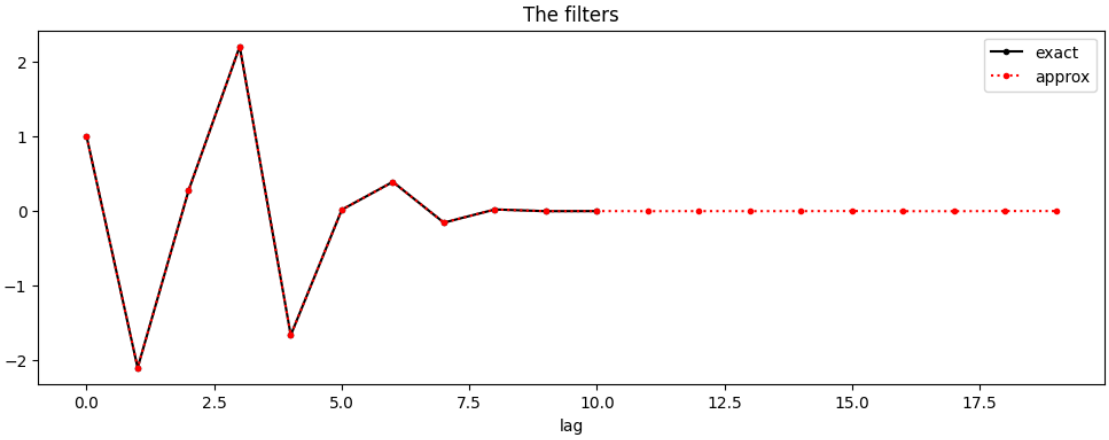


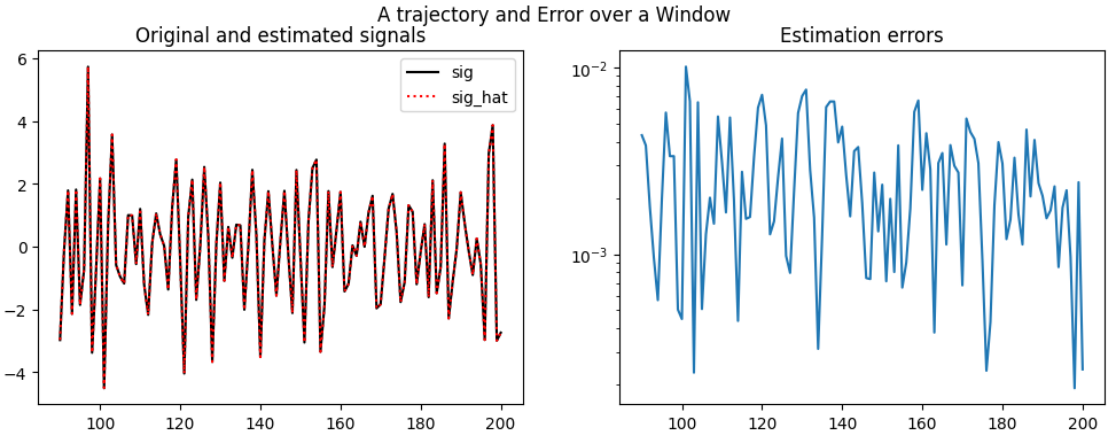

```
MSE: 6.479685687494901e-6
Time: 1.00049044 sec
```

1.3 CKMS+ (1 iteration)

[24]:
```
# 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: 3.410320241
Bytes Allocated: 8859425298
Time taken for spectfact: 0.228043221
Bytes Allocated: 173515269
```



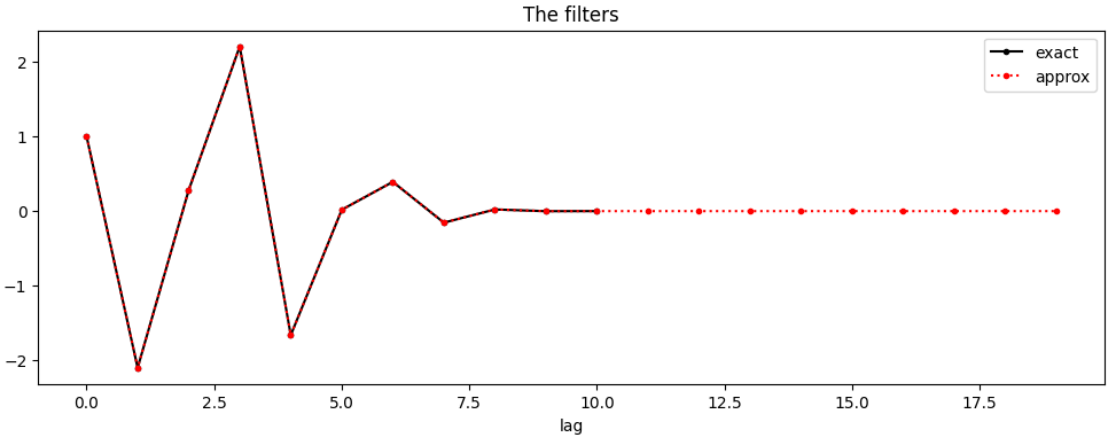


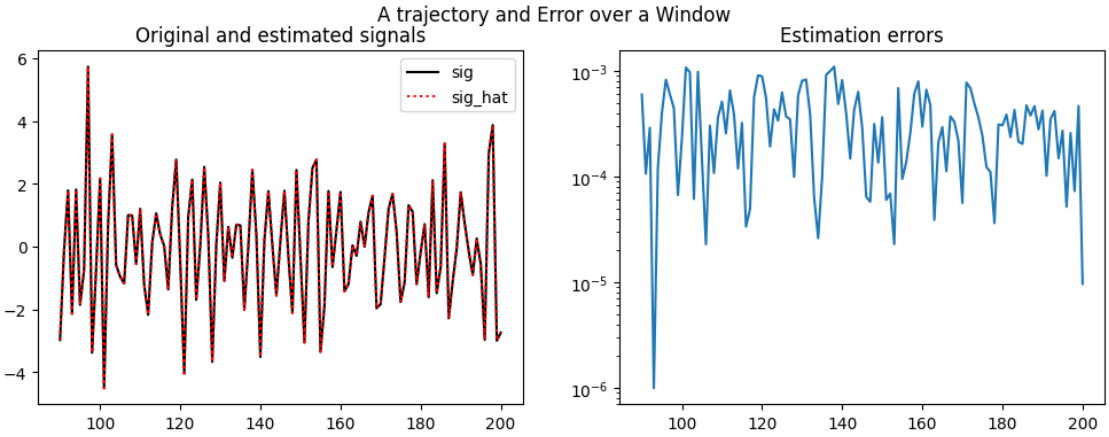

```
MSE: 4.664729220932354e-6
Time: 3.793853966 sec
```

1.4 CKMS+ (2 iterations)

[25]:
```
# 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: 8.134001766
Bytes Allocated: 23151236128
Time taken for spectfact: 0.205059423
Bytes Allocated: 373337232
```






```
MSE: 7.936565516756473e-8
Time: 8.422552173 sec
```