**Wilson-Cowan Model**

Şeyma Bayrak

**Abstract**

Wilson-Cowan model can be solved numerically, but there is a small issue to consider; since the model depends on General Logistic Function, one must consider about beginning with calculation of GLF. This document starts with GLF observation with different inputs – s. The further purpose is to create more sophisticated input and then calculate the corresponding u(t) as analytically shown in Wilson-Cowan model. For each different s, GLF is found, then inserted into numerical integration loop for u(t). This document presents a ramp input, a flat input and finally a square wave input for GLF, and shows how u(t) looks for each of the input.

GLF depends on parameters “alpha,beta,nu” and a variable “x”. Beta and nu are chosen to be constant, while alpha is chosen as depending on stimulus s.

**View GLF with several inputs –s.**

s=linspace(0,1,5);

s = 0 0.2500 0.5000 0.7500 1.0000

function y=GLFview(x,s, params)

params.NU=1;

params.BETA=6;

params.ALPHA0=3.6;

params.ALPHA1=1.8;

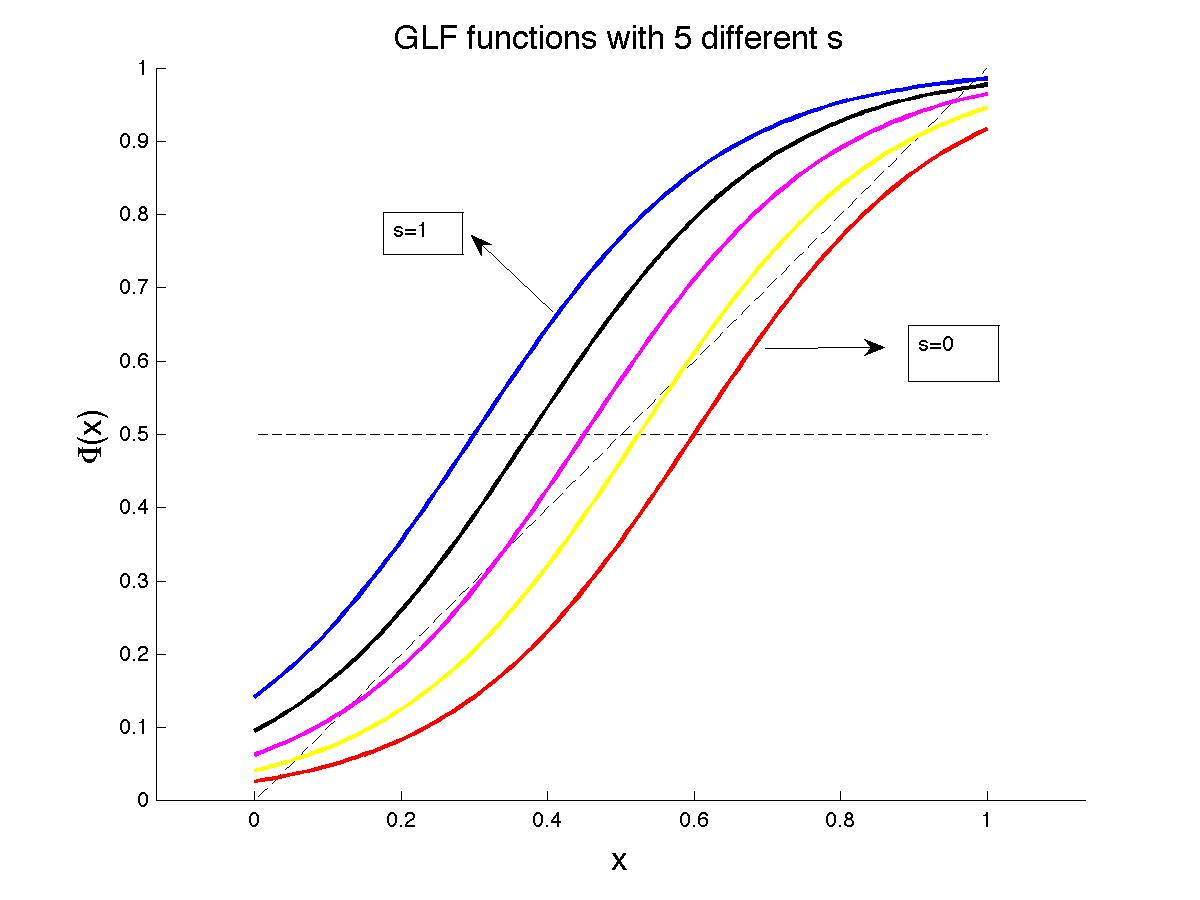
% make alpha depended on input s

alpha=params.ALPHA0 + (params.ALPHA1 - params.ALPHA0)\*s;

% now express GLF function

y=(1+exp(-params.BETA\*x+alpha)).^(-1/params.NU);

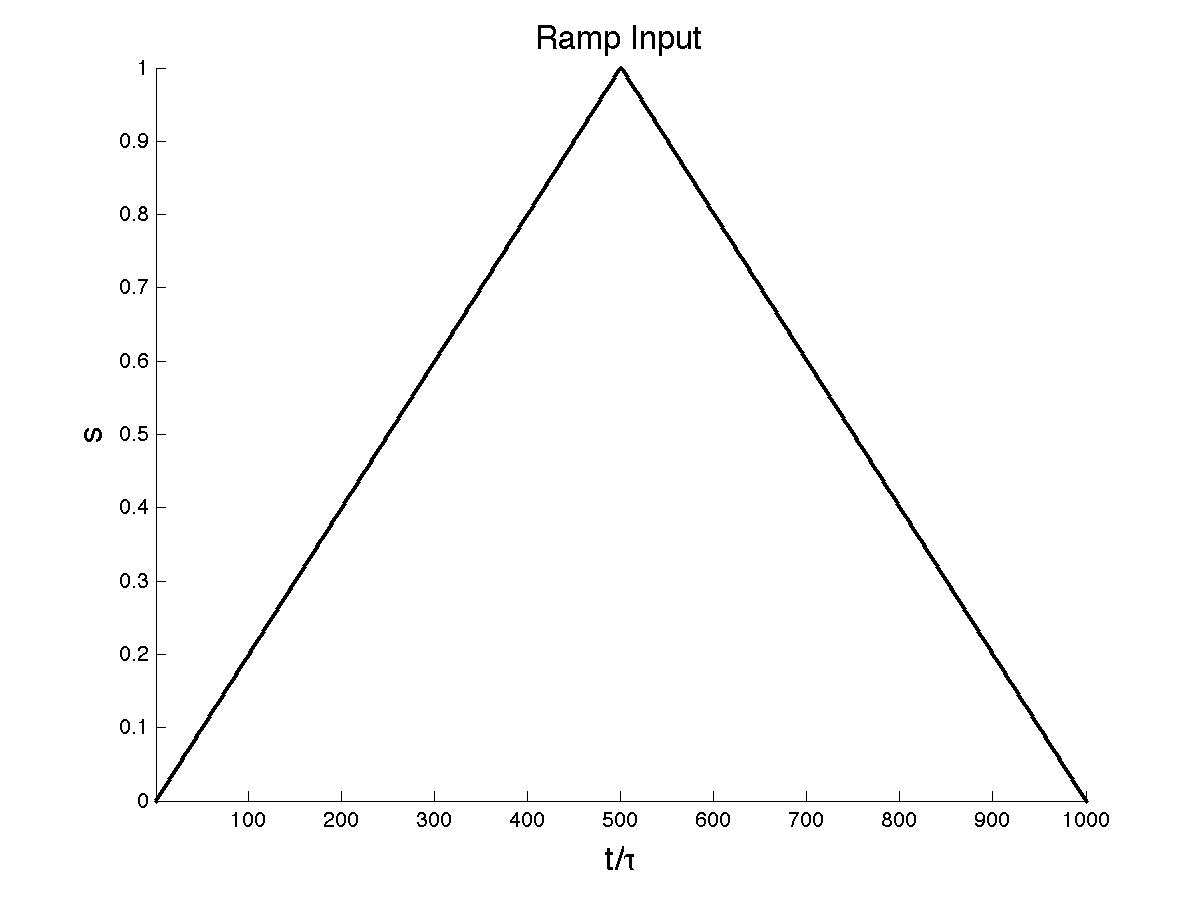
return;



Alpha increases with s, and GLF shifts to the left.

**Create a ramp input as**

st=[linspace(0,1,n\_step) linspace(1,0,n\_step)];



**Wilson-Cowan Model**

Solve u (I wrote exactly u inside the function but later on called it as xt)

function u=WilCow(st,x0,dt,TAU,SIGMA)

% Wilson-Cowan Model, numerical integ. to solve u

dt=dt/TAU;

u=zeros(1,numel(st));

u(1)=x0;

nt = SIGMA \* sqrt(dt) \* normrnd(0,1,size(st));

for i=2:numel(st)

du= dt \* (-u(i-1) + GLFview ( u(i-1), st(i-1) ) ) + nt(i-1);

u(i)=u(i-1)+du;

end

What does the command “normrnd” mean ?

Wilson-Cowan model has an additional Gaussian White Noise: . Further math introduces this term as . The term “n(t)” is chosen as a random number from a normal distribution with mean 0 and standard deviation 1.

*NORMRND Random arrays from the normal distribution.*

*R = NORMRND(MU,SIGMA) returns an array of random numbers chosen from a*

*normal distribution with mean MU and standard deviation SIGMA.*

x0=0.3;

Tend=5;

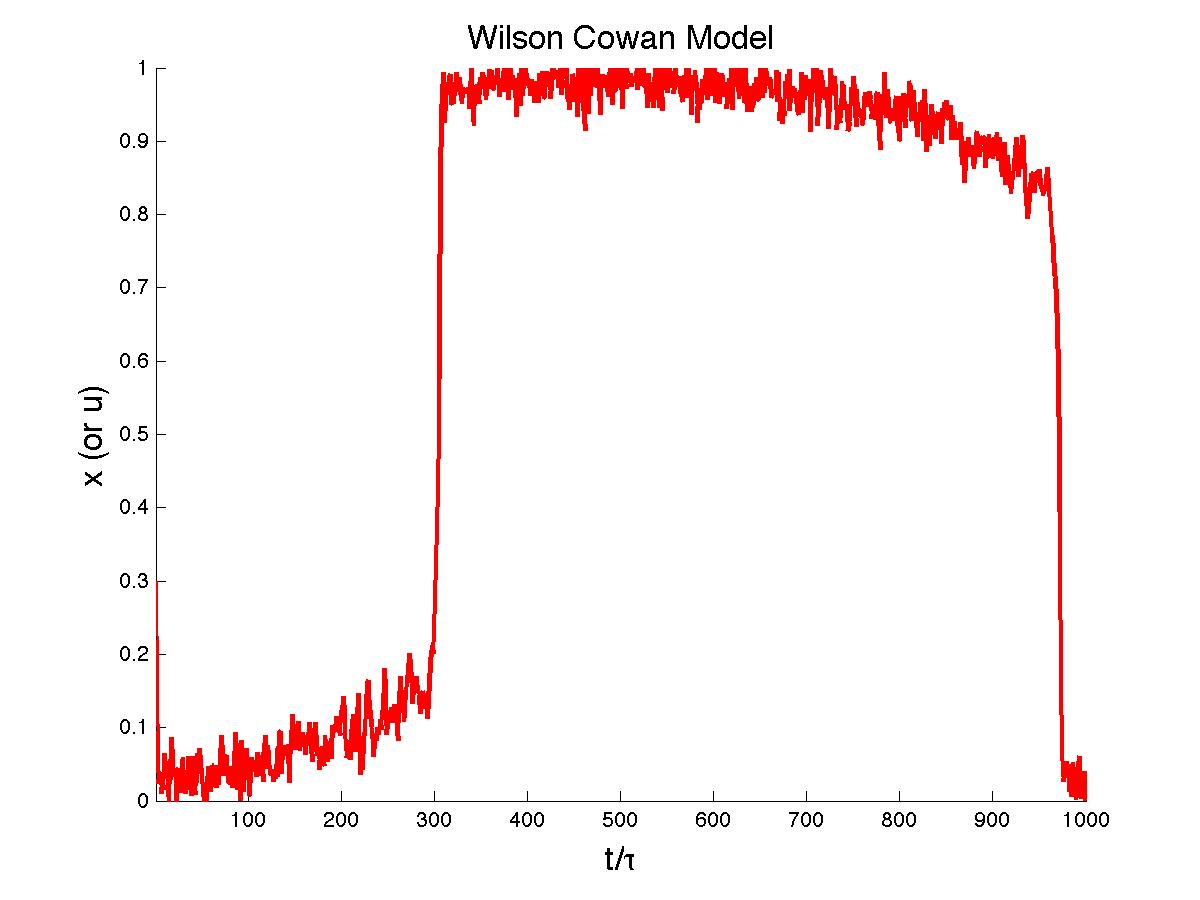
n\_step=Tend/DT;

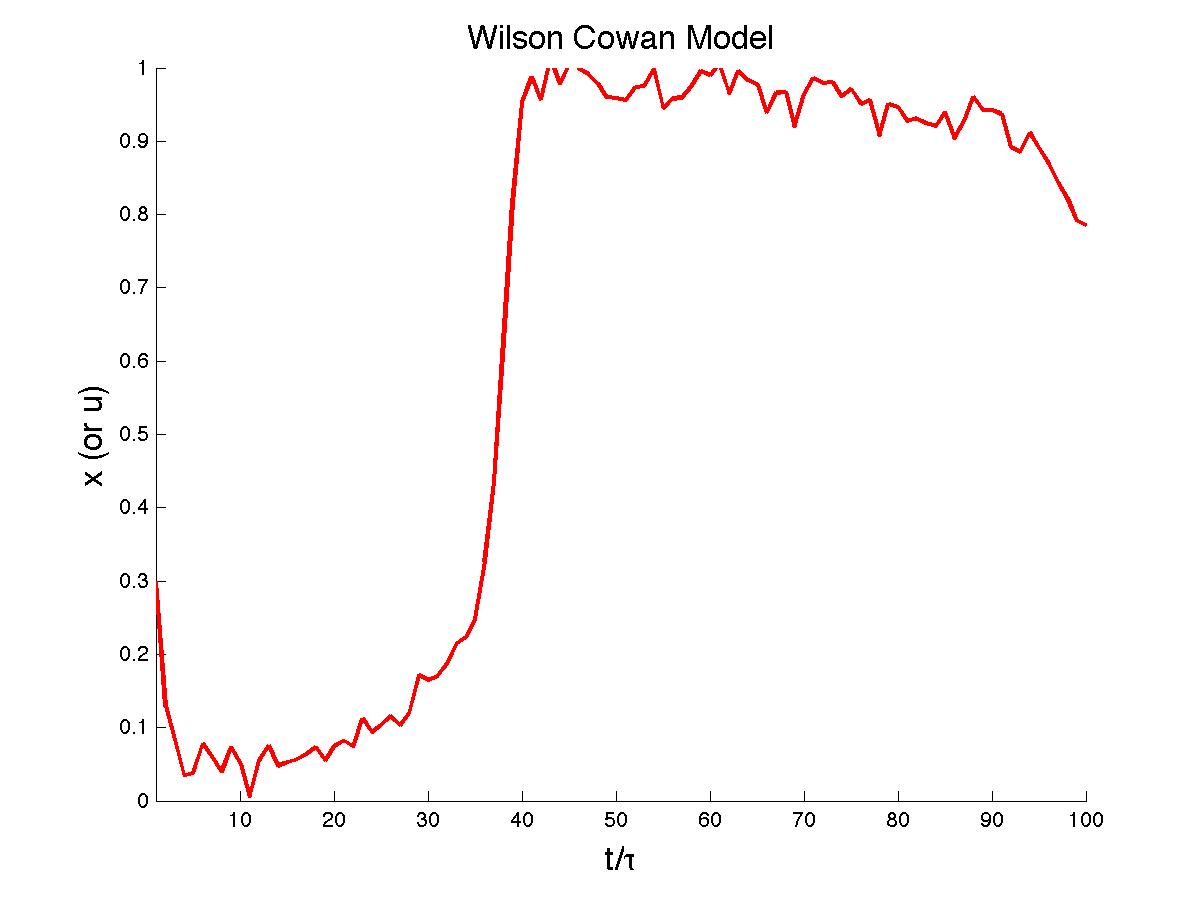
st=[linspace(0,1,n\_step) linspace(1,0,n\_step)];

tt = DT \* (1:numel(st)) / TAU;

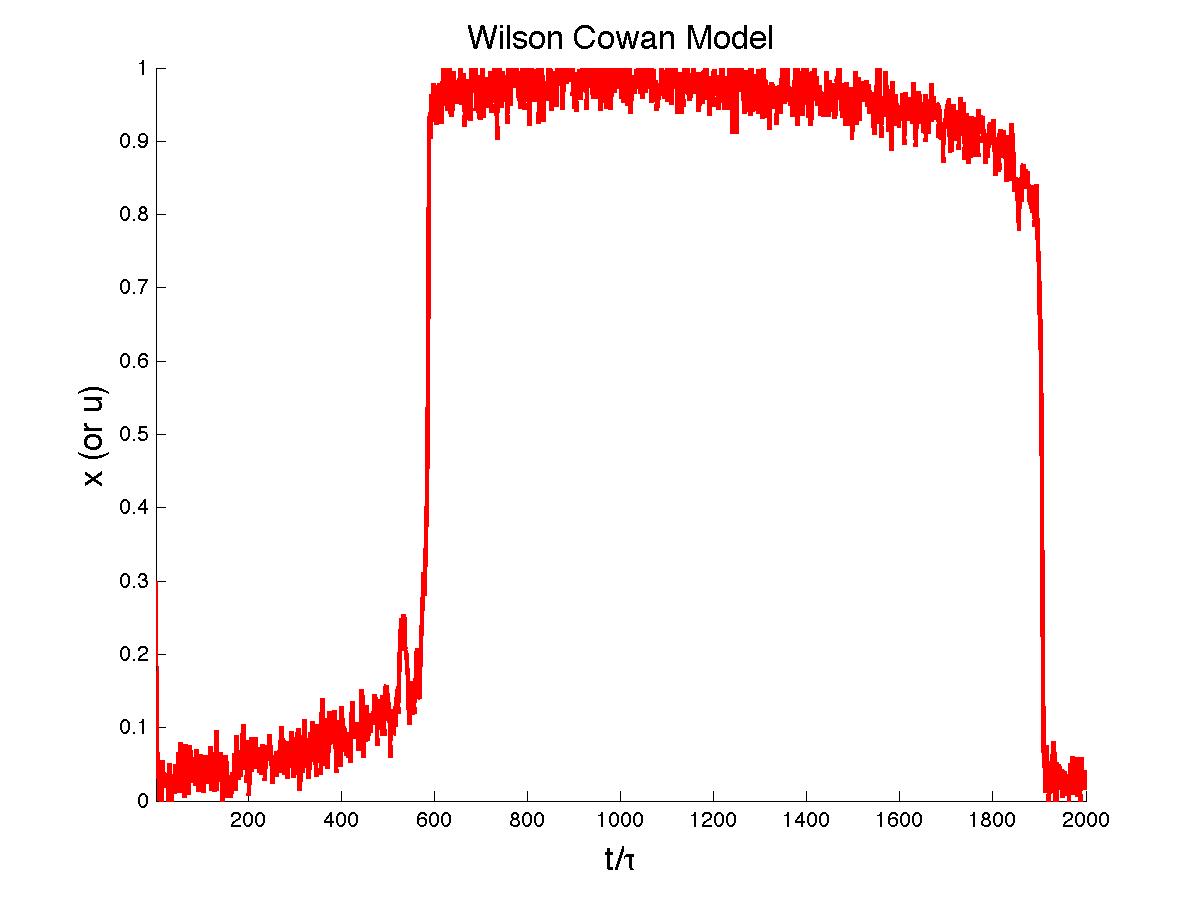
xt =WilCow( st, x0, DT, TAU, SIGMA );

plot(tt,xt,'r', 'LineWidth',2);

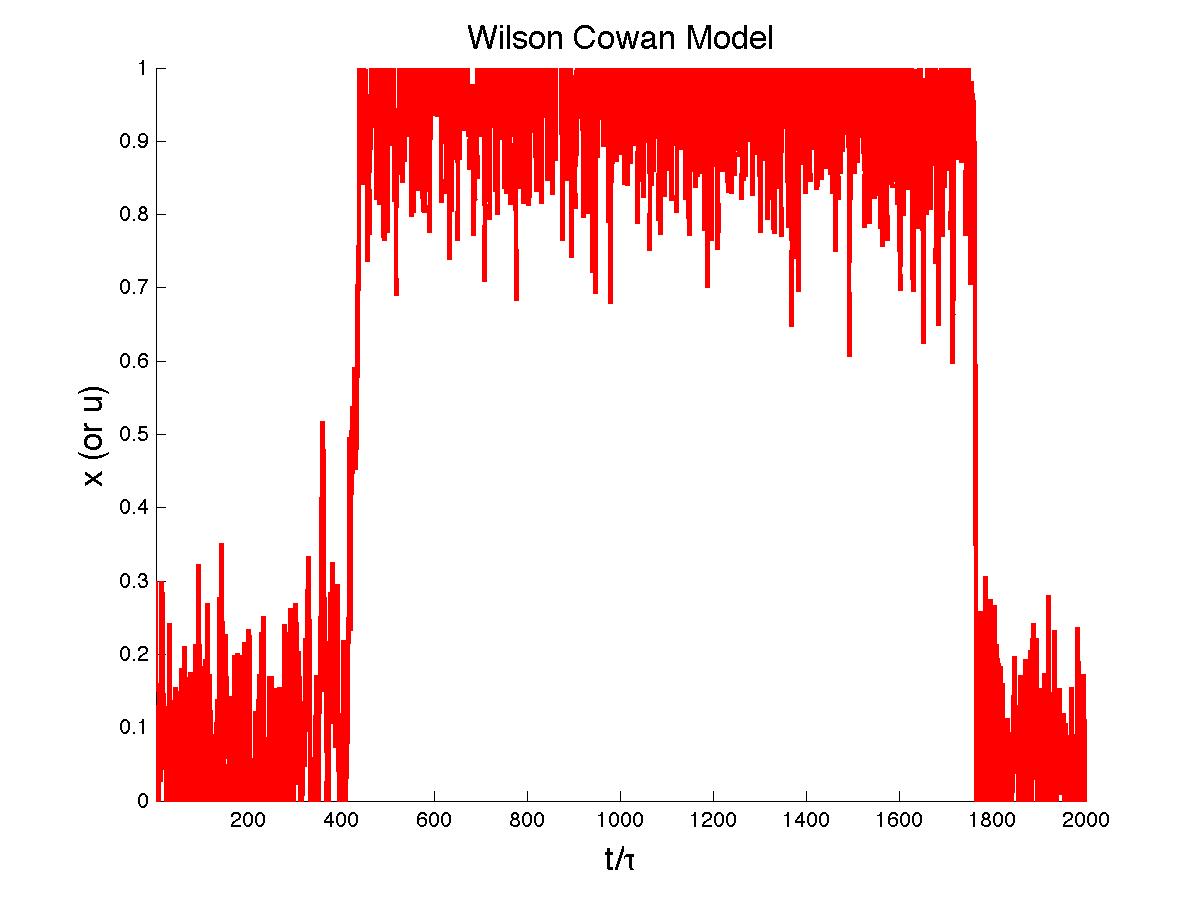


Change T from T=5 to t=0.5 >>>>>>>

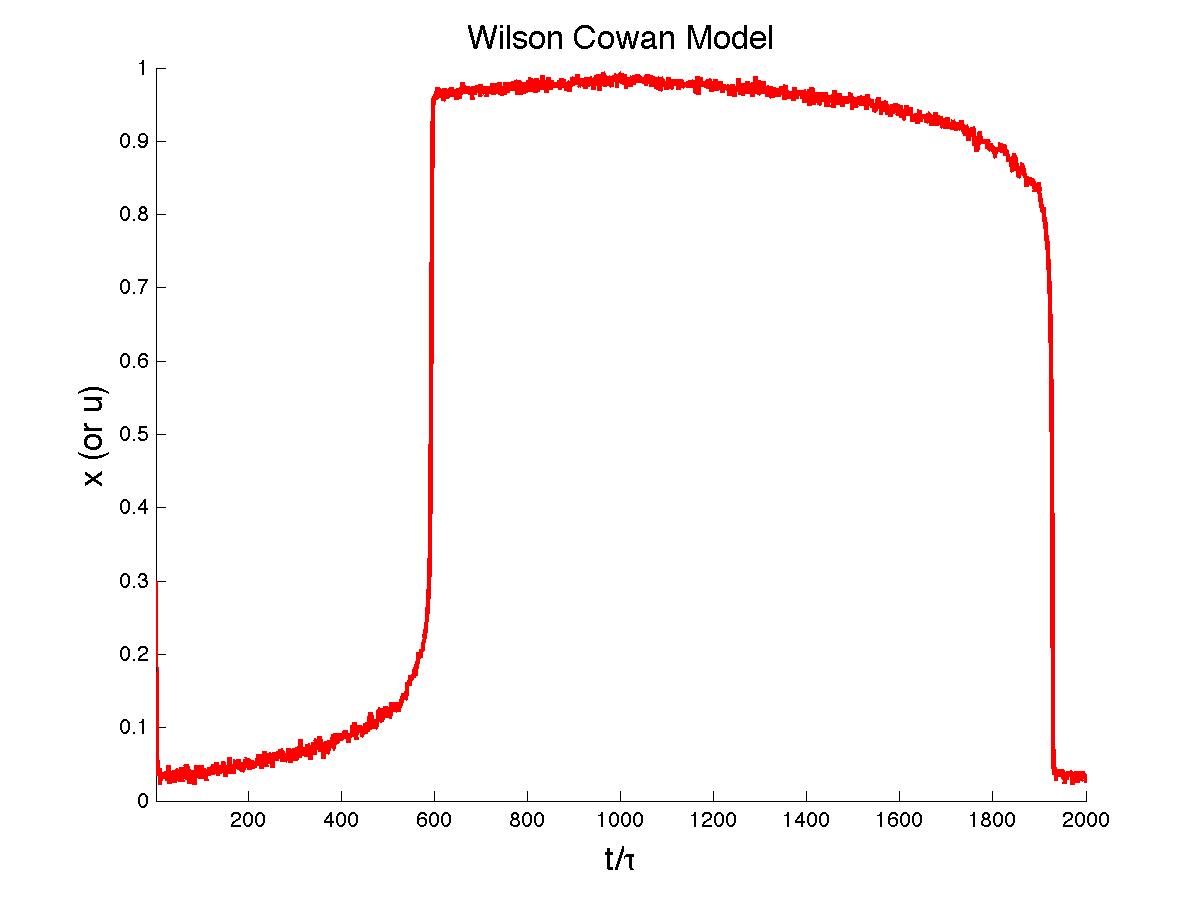
Change T from T=5 to T=10 >>>>>>>>



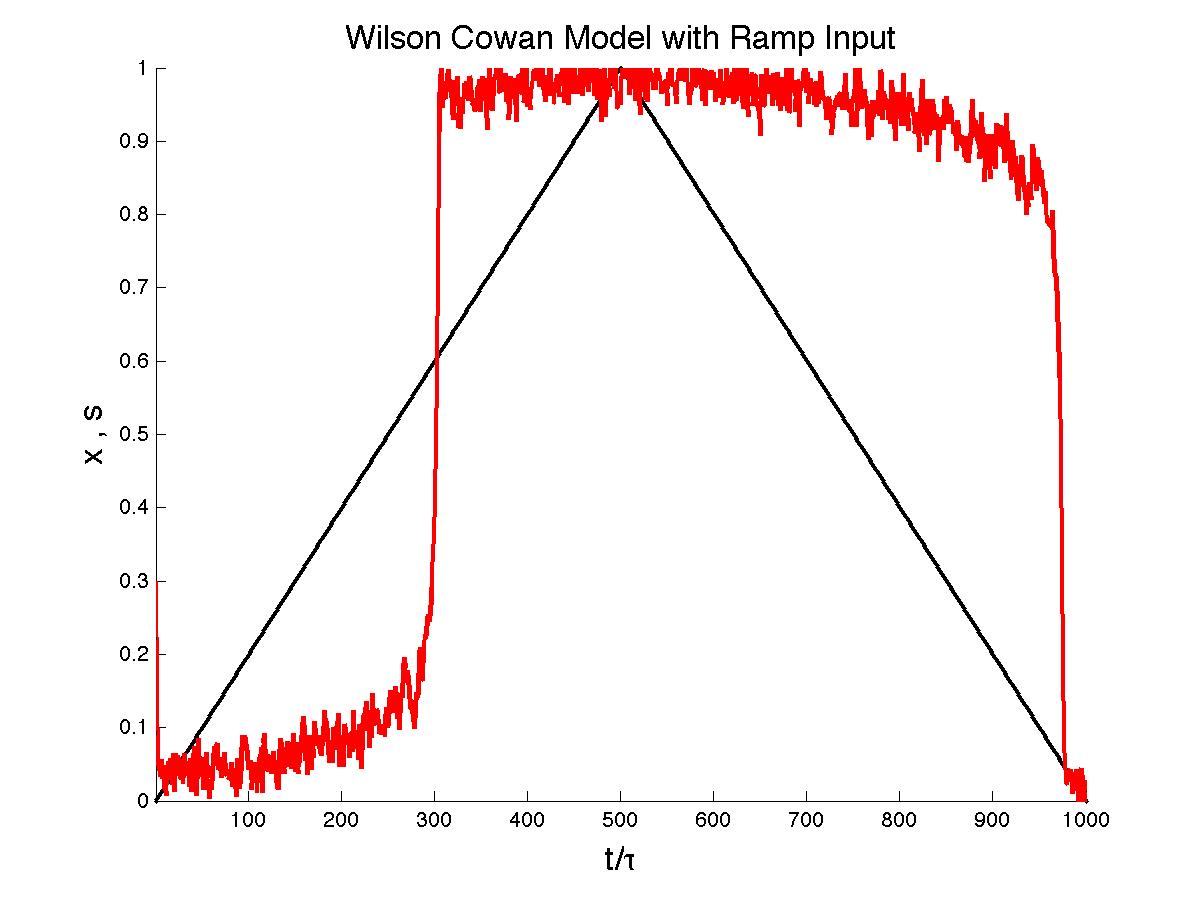
Note that sigma ) is the noise amplitude. Let us change it from =0.02 to =0.1 >>>



Let us change it from =0.02 to =0.05 >>>>>>>

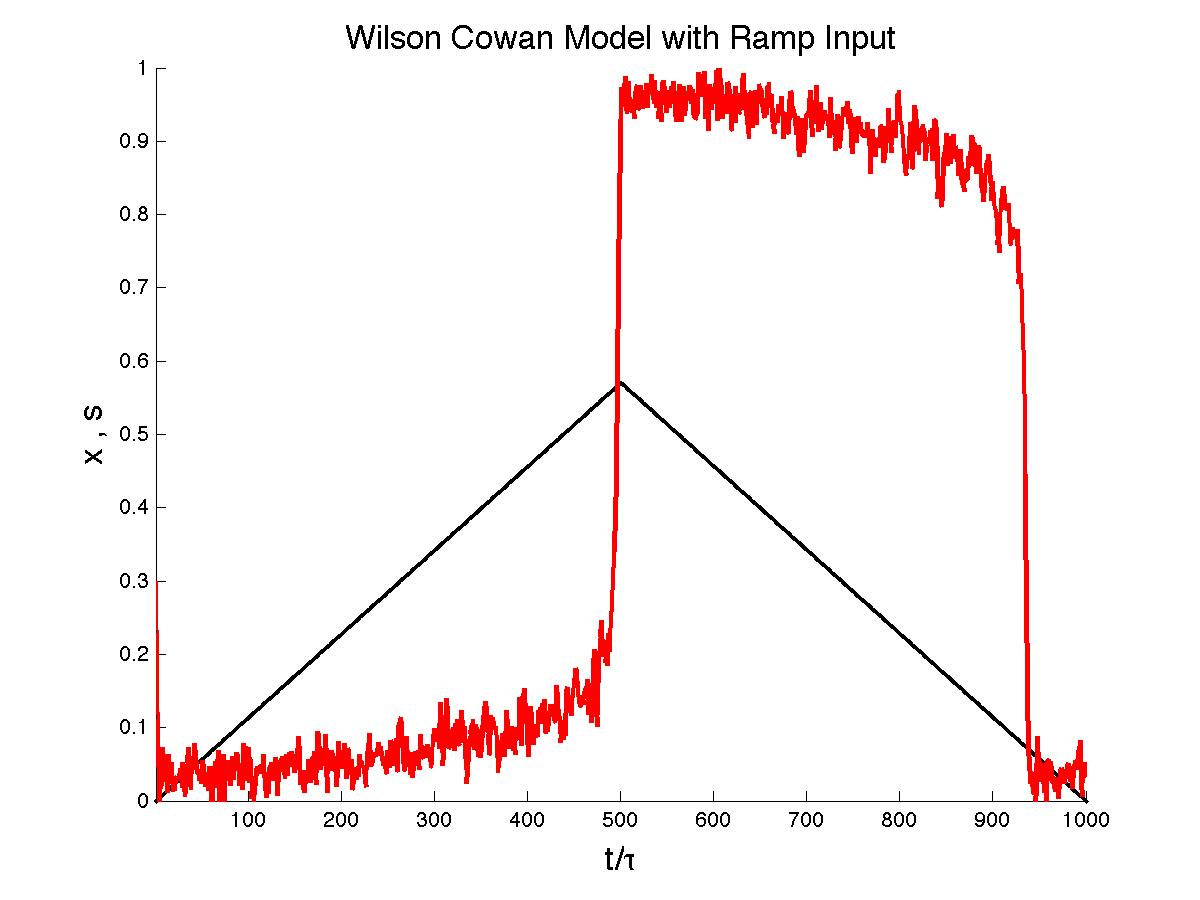


Now let us plot both ramp input s and x (which is found out by numerical integration of Wilson Cowan Method) on the same graph.



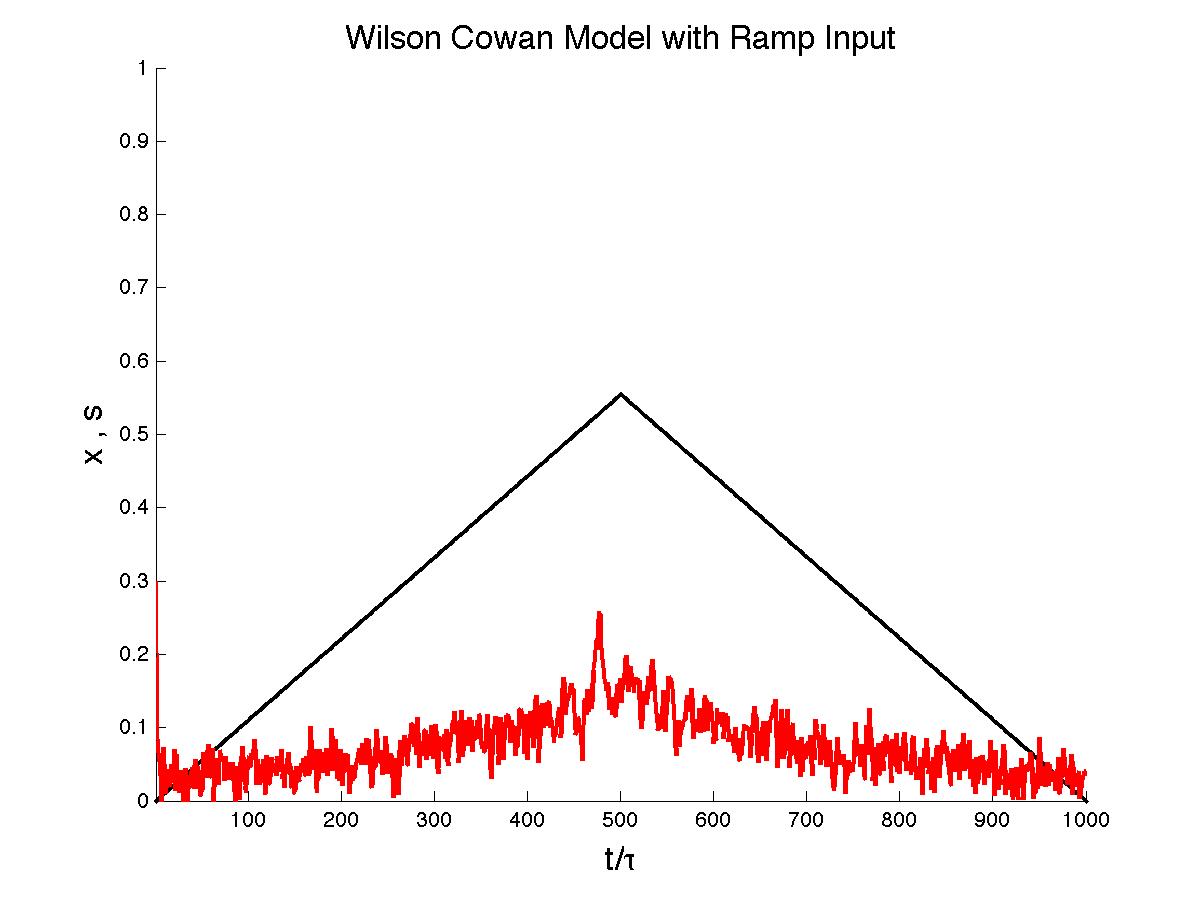
What happens if I decrease the stimulus strength such that (instead of linspace(0,1,n\_step)) to:

st=[linspace(0,0.6,n\_step) linspace(0.6,0,n\_step)];



If I decrease the strength further such that:

st=[linspace(0,0.554,n\_step) linspace(0.554,0,n\_step)];



**Steady Sate Points**

What does “steady state” point means? In my first exercise namely GlogF.m, I had plotted the x=y line in almost all graphs. Steady state points are where the sigmoidal GLF function is extremely close to the x=y line (ideally intersection points.) This new exercise has the same logic to find out steady state points.

Logic: Find the roots where FUN(x,a,b)x, MATLAB solves it as the following:

1. Define another function such as NEW\_FUN(x,a,b)=GLF(x,a,b)-x
2. x\_steady\_state=fsolve(@(x) NEW\_FUN(x,a,b) )
3. Guess where and how many steady state we could have

Since one sigmoidal GLF can intersect maximum 3 times with x=y line:

1. x\_steady\_state=fsolve(@(x) NEW\_FUN(x,a,b), [0.01 0.5 0.99] )

Note that , so it is the best to guess roots at beginning, intermediate and end levels.

function [yss\_hi, yss\_mid, yss\_lo]=WilCowSS(st)

yss\_hi = nan(size(st));%last ss point of one GLF (sigmoid)

yss\_lo = nan(size(st));%first ss point of one GLF (sigmoid)

yss\_mid =nan(size(st));%intm. ss point of one GLF (sigmoid)

params.NU=1;

params.BETA=6;

params.ALPHA0=3.6;

params.ALPHA1=1.8;

for i=1:numel(st)

xss=fsolve(@(x) GLFviewSS(x, st(i)), [0.01 0.5 0.99],optimset('Display','off'));

yss=GLFview(xss, st(i), params);

if abs(yss(1)-xss(1)) < 0.01

yss\_lo(i) = yss(1);

end

if abs(yss(2)-xss(2)) < 0.01

yss\_mid(i) = yss(2);

end

if abs(yss(3)-xss(3)) < 0.01

yss\_hi(i) = yss(3);

end

end

return;

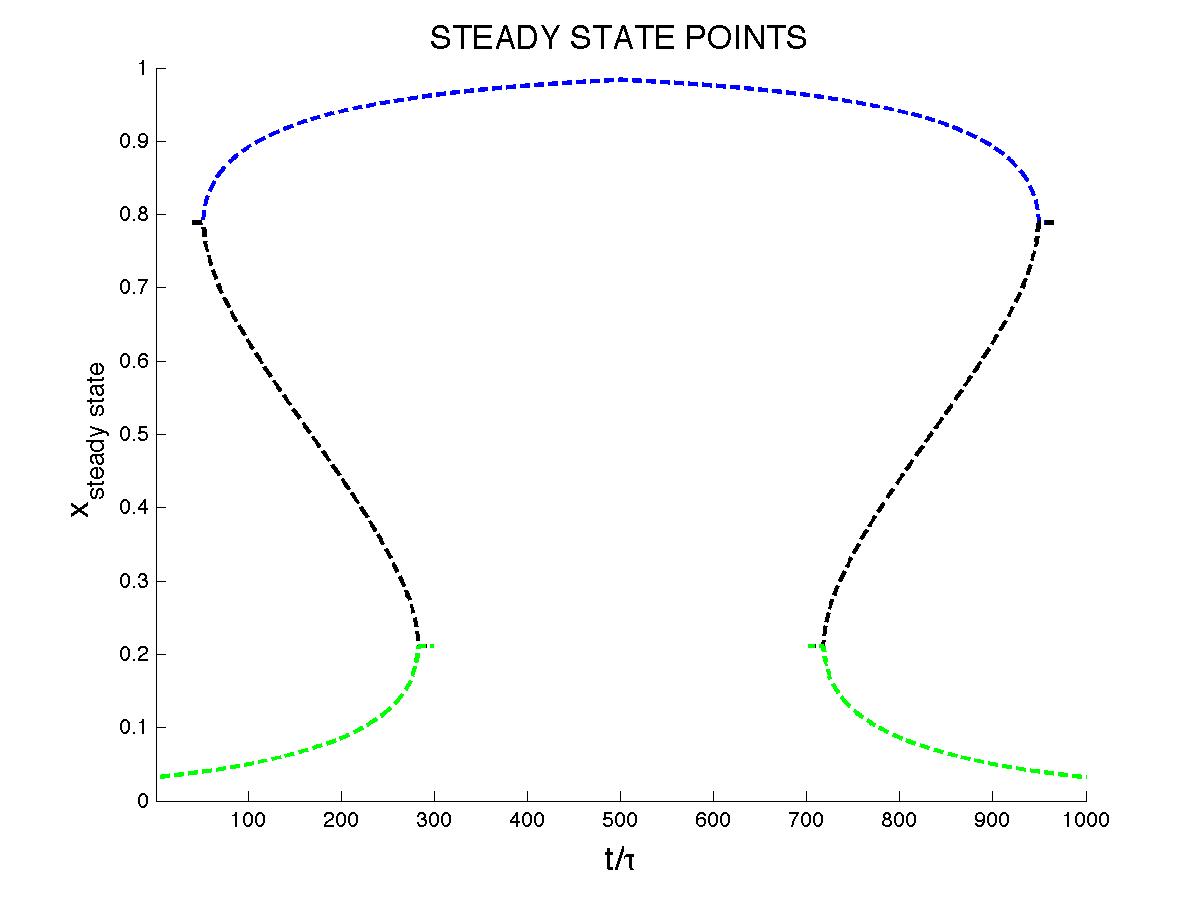
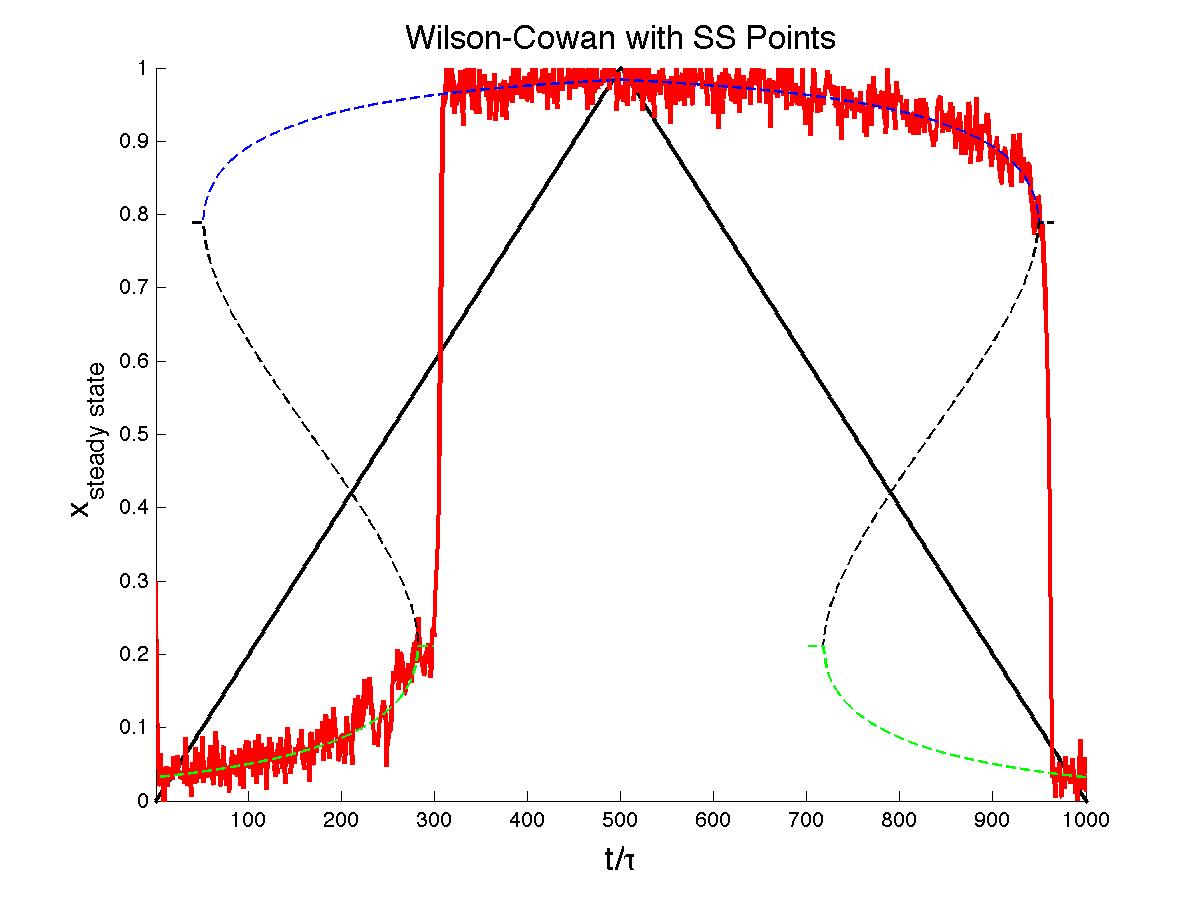
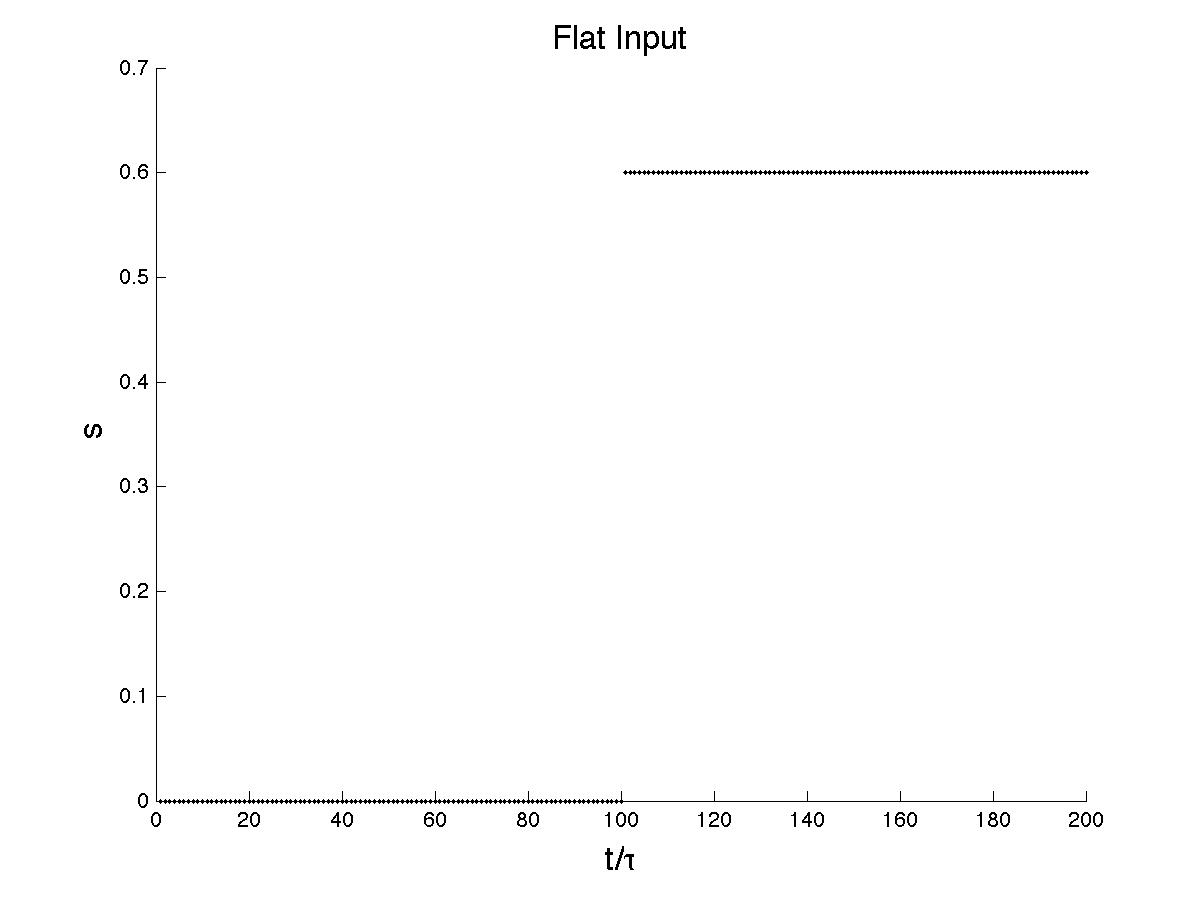


FIGURE: Steady state points of many GLF (bcs of different s) for the RAMP INPUT

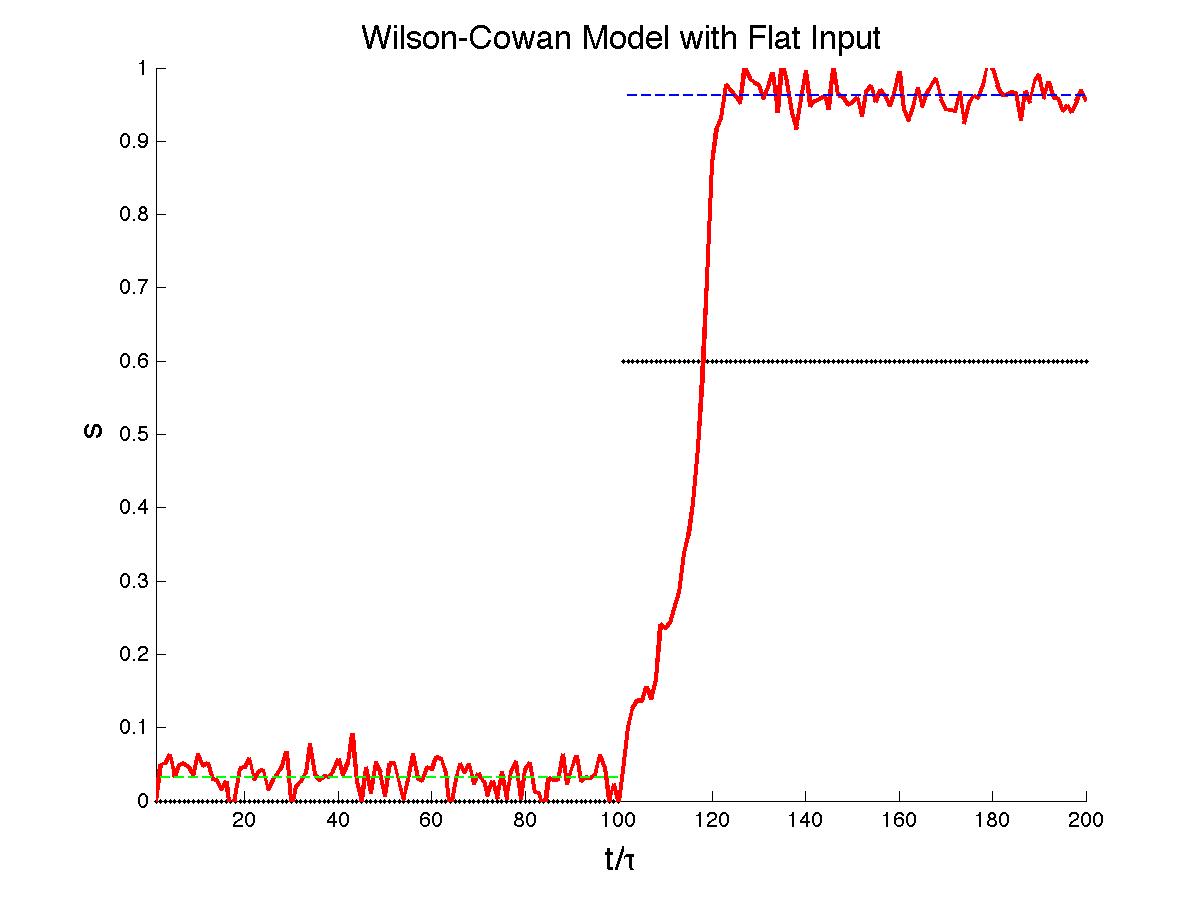
Let us plot ramp input s, Wilson-Cowan x(t) (which of course depends on GLF), and steady state points on the same graph.



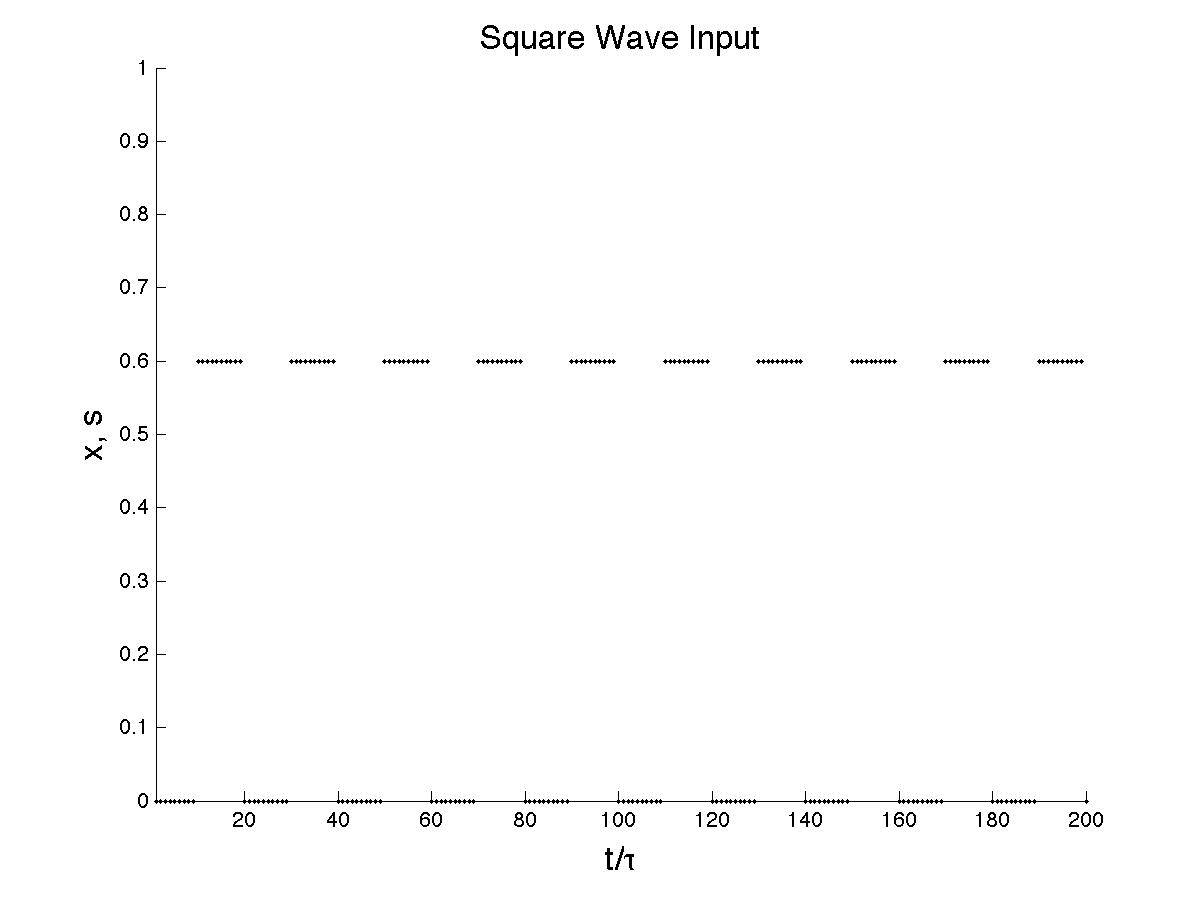
**Flat Input >>>>** st=[s0\*ones(1,nstep) s1\*ones(1,nstep)];



How does x(t) and steady state pints look like with flat input?



**Square Wave Input**



How does Wilson-Cowan look like with the input s of square waves?

