Conclusion

Code for Stochastic GW Background

QYQ

2017.4.21

Recently,I use the algorithm *Simulator 4* to simulate the Stochastic Gravitational Wave Background(SGWB) and at here I will demonstrate the algorithm and functions used in the *simulator 4*.

---

1. The Creation of Random GW Sources

1.1 GenerateRandomGWSource

In this algorithm I use the function *GenerateRandomGWSource* to create the random distribution of various parameters which Ire used to identify the properties of the GW sources.

```
[Amp,alpha,delta,fgw,iota,Psi,Phi0,r]=GenerateRandomGWSource(Ns)
```

Where Amp, alpha, delta, fgw, iota, Psi, Phi0, r are the sources' amplitude, longitude, altitude, frequency, iota, Psi, Phi0, the distance from source to us and Ns is the number of sources.
In this algorithm, I assume the sources are uniformly distributed in the space, and this requires the distribution intensity of sources should be appropriate to the r^2. At first I create a distribution appropriate to r^3 uniformly during the range (0,1) then find the distribution of its cube roots and I get the distribution of r^2.

```
pr0 = random('uniform',0,1,1,Ns);
pr1 = pr0.^(1/3);
histgram(Pr1)
```

![Pr分布](/Users/qianyiqian/Desktop/MatlabPrograms/PTAcode/Conclusion/Pr.jpg
)
As for the other patameters:

```
log10Mc = random('uniform',6,10,1,Ns);
Mc = 10.^(log10Mc);
[alpha,delta]=SpherePointPicking(Ns);
log10fgw = random('uniform',-9,-6,1,Ns);% generate uniform distribution of log
fgw = 10.^(log10fgw);% get the real fgw
iota = random('uniform',0,pi,1,Ns);
Psi = random('uniform',0,pi,1,Ns);
Phi0 = random('uniform',0,pi,1,Ns);
```

where Mc is the Chirp Mass is uniformly distributed in $(10^6,10^9)$, fgw is distributed uniformly in $(10^{-9},10^{-6})$, iota, Psi, Phi0 are all distributed uniformly in $(0,\pi)$. Where the amplitude of the GW source
$$Amp=\frac{G\mu a^2\omega^2}{c^4D}=\frac{G^{5/3}}{c^4D}M_c^{5/3}\omega^{-1/3}$$

1.2 SpherePointPicking

There is another function in this algorithm *SpherePointPicking*. This function is used to take some points in the sphere randomly and these points are used as the random locations of GW sources.

```
function [theta,phi]=SpherePointPicking(n)
%Uniform random Sphere Point Picking
%r = 1;
NN = n;
u = random('uniform',0,1,1,NN);
v = random('uniform',0,1,1,NN);
theta = 2*pi*u;
phi = acos(2*v-1)-pi/2;
```

2. Pulsar Timing Arrays

To detect the GW, I can exploit the instinctive precise clocks, pulsars, to detect the very small degree of the signals of GW. To do this, I need to calculate the timing residuals brought by the effects of GW. To use these pulsars, I need to know their locations in the sky and the distance to us. In this algorithm I use 17 pulsars to do so.

```
tmp1='J0030+0451';
alphaP(1)=(0*15+30*15/60)*pi/180;
deltaP(1)=(4+51/60)*pi/180;
distP(1)=1.376*kilo*pc2ly;  % in ly
sd(1)=1.0*10^(-8); %0.148;

tmp2='J0613-0200';
alphaP(2)=(6*15+13*15/60)*pi/180;
deltaP(2)=-(2+0/60)*pi/180;
distP(2)=6.318*kilo*pc2ly;
sd(2)=1.0*10^(-8); %0.178;

tmp3='J1713+0747';
alphaP(3)=(17*15+13*15/60)*pi/180;
deltaP(3)=(7+47/60)*pi/180;
distP(3)=7.524*kilo*pc2ly;
sd(3)=1.0*10^(-8); %0.03;

tmp4='J1909-3744';
alphaP(4)=(19*15+9*15/60)*pi/180;
deltaP(4)=-(37+44/60)*pi/180;
distP(4)=3.532*kilo*pc2ly;
sd(4)=1.0*10^(-8); %0.038;

%-----------
tmp5='J1012+5307';
alphaP(5)=(10*15+12*15/60)*pi/180;
deltaP(5)=(53+7/60)*pi/180;
distP(5)=1.045*kilo*pc2ly;
sd(5)=1.0*10^(-8); %0.276;

tmp6='J1455-3330';
alphaP(6)=(14*15+55*15/60)*pi/180;
deltaP(6)=-(33+30/60)*pi/180;
distP(6)=6.593*kilo*pc2ly;
sd(6)=1.0*10^(-8); %0.787;

tmp7='J1600-3053';
alphaP(7)=(16*15+0*15/60)*pi/180;
deltaP(7)=-(30+53/60)*pi/180;
distP(7)=13.532*kilo*pc2ly;
```

```
sd(7)=1.0*10^(-8); %0.163;

tmp8='J1640+2224';
alphaP(8)=(16*15+40*15/60)*pi/180;
deltaP(8)=(22+24/60)*pi/180;
distP(8)=3.675*kilo*pc2ly;
sd(8)=1.0*10^(-8); %0.409;

% ---- add other nanograv pulsars  09/25/2014  YW
tmp9='J1643-1224';
alphaP(9)=(16*15+43*15/60)*pi/180;
deltaP(9)=-(12+24/60)*pi/180;
distP(9)=2.735*kilo*pc2ly;
sd(9)=1.0*10^(-8);

tmp10='J1744-1134';
alphaP(10)=(17*15+44*15/60)*pi/180;
deltaP(10)=-(11+34/60)*pi/180;
distP(10)=0.453*kilo*pc2ly;
sd(10)=1.0*10^(-8);

tmp11='J1853+1308';
alphaP(11)=(18*15+53*15/60)*pi/180;
deltaP(11)=(13+8/60)*pi/180;
distP(11)=7.243*kilo*pc2ly;
sd(11)=1.0*10^(-8);

tmp12='B1855+09';  % J1857+0943
alphaP(12)=(18*15+57*15/60)*pi/180;
deltaP(12)=(9+43/60)*pi/180;
distP(12)=3.239*kilo*pc2ly;
sd(12)=1.0*10^(-8);

tmp13='J1910+1256';
alphaP(13)=(19*15+10*15/60)*pi/180;
deltaP(13)=(12+56/60)*pi/180;
distP(13)=13.542*kilo*pc2ly;
sd(13)=1.0*10^(-8);

tmp14='J1918-0642';
alphaP(14)=(19*15+18*15/60)*pi/180;
deltaP(14)=-(6+42/60)*pi/180;
distP(14)=6.437*kilo*pc2ly;
sd(14)=1.0*10^(-8);

tmp15='B1953+29';  % J1955+2908
alphaP(15)=(19*15+55*15/60)*pi/180;
deltaP(15)=(29+8/60)*pi/180;
distP(15)=0.875*kilo*pc2ly;
sd(15)=1.0*10^(-8);

tmp16='J2145-0750';
alphaP(16)=(21*15+45*15/60)*pi/180;
```

```
deltaP(16)=-(7+50/60)*pi/180;
distP(16)=3.982*kilo*pc2ly;
sd(16)=1.0*10^(-8);

tmp17='J2317+1439';
alphaP(17)=(23*15+17*15/60)*pi/180;
deltaP(17)=(14+39/60)*pi/180;
distP(17)=5.281*kilo*pc2ly;
sd(17)=1.0*10^(-8);
```

3. Calculation of the Timing Residulas

In *Simulator 4*, I accumulate the timing residuals brought by every sources for each pulsar and I will get final timing residuals（each pulsar）：

Detail calculation for timing residual is in the function *FUllResiduals*.

```
for i=1:1:Np
    for j=1:1:Ns  % number of GW sources

        % GW sky location in Cartesian coordinate
        k=zeros(1,3);  % unit vector pointing from SSB to source
        k(1)=cos(delta_tmp(j))*cos(alpha_tmp(j));
        k(2)=cos(delta_tmp(j))*sin(alpha_tmp(j));
        k(3)=sin(delta_tmp(j));
        theta=acos(k*kp(i,:)');
        %sprintf('%d pulsar theta=%g',i,theta)
        %phiI(i)=mod(phi0-omega*distP(i)*(1-cos(theta)), 2*pi);  % modulus aft
        %phiI(i)=mod(2*phi0-omega_tmp(l)*distP(i)*(1-cos(theta)), pi);  % modu
        phiI(i)=mod(phi0(j)-0.5*omega_tmp(j)*distP(i)*(1-cos(theta)), pi);  %

        tmp = FullResiduals(alpha_tmp(j),delta_tmp(j),omega_tmp(j),phi0(j),phi
            Amp(j),iota(j),thetaN(j),theta,yr);
        timingResiduals_tmp(i,:) = timingResiduals_tmp(i,:)+tmp';

    end

    inParams = struct('Np',Np,'N',N,'Ns',Ns,'s',timingResiduals_tmp,'sd',sd,..
        'alphaP',alphaP,'deltaP',deltaP,'kp',kp,'yr',yr);


end
```

Plotting the first pulsar's timingResiduals.
`![timingResiduals_1](/Users/qianyiqian/Research/Play Ground/PTAcode/Conclusion/timingResidual_1.jpg)`

```
plot(timingResiduals_tmp(1,:))
```

4. Calculation for Correlation Coefficient

After I get the timing residuals, I need to calculate the correlate coefficient for every two pulsars, and I can use this equation:
$$r(\theta)=\frac{1}{N}\sum^{N-1}_{i=0}R(t_i,\hat {k_1})R(t_i,\hat {k_2})$$
where $R(t_i,\hat {k_1})$is the timing residual for time i , $\hat {k_1}$ is the first pulsar's location vector to us.
After I match every two pulsars as a pair, I calculate the correlation coefficient for every pair and I get：

```
function [CE,thetaC] = CorrelationCoefficient()
run simulator4.m;
load('/Users/qianyiqian/desktop/matlabprograms/PTAcode/GWB/GWB.mat');
Np=getfield(simParams,'Np');
kp=getfield(simParams,'kp');
%N=getfield(simParams,'N');
cthetaC=zeros(1,(Np-1)*Np/2);%%%cos(theta) between every two pulsar
CE=zeros(1,(Np-1)*Np/2);%%%correlation coefficients
ct = 1;%% counter

for i=1:1:Np-1
    for j=1+i:1:Np

        cthetaC(:,ct)=kp(i,:)*kp(j,:)';%%% cos(theta) between every two pulsar
        R=(timingResiduals_tmp(i,:)*timingResiduals_tmp(i,:)')...
       *(timingResiduals_tmp(j,:)*timingResiduals_tmp(j,:)');
        CE(:,ct) = timingResiduals_tmp(i,:)*timingResiduals_tmp(j,:)'/sqrt(R);
        ct = ct+1;

    end
end
thetaC=acos(cthetaC)*180/pi;%%%% theta between every two pulsar
%plot(thetaC,CE,'.k');
```

5.Helling-Downs Curve

Run the *Simulator 4* and *CorrelationCoefficient* function repeatedly, I can get lots of different correlation coefficients for every pairs, because the parameters used to determine the sources are random and at last I simply average these coefficients and I can get the Helling-Downs Curve perfectly fit the theoretical prediction:

$$\xi(\theta)=\frac{3}{2}x\log x -\frac{x}{4}+\frac{1}{2}$$

$$x=[1-\cos (\theta)]/2$$