LECTURE 7: FUNDAMENTALS OF OBSERVATIONAL COSMOLOGY (III)

GONG-BO ZHAO

1. Parameter estimation using SN Ia data

As we covered in last lecture, we can minimise the χ^2 in order to perform parameter estimations.

Suppose we want to constrain $\Omega_{\rm M}$ and Ω_Λ using SN Ia data (we fix H_0 for simplicity), then we can minimise the following χ^2,

$$\chi^2(\Omega_{\rm M}, \Omega_\Lambda) = \Delta^T C^{-1} \Delta \tag{1}$$

where the column vector Δ is,

$$\Delta \equiv \vec{\mu}_{\rm theo.} - \vec{\mu}_{\rm obs.} \tag{2}$$

The code we wrote together in class is attached in the Appendix.

2. The Fisher Information Matrix

The Fisher matrix technique is widely used in cosmology for performing forecasts on the uncertainty of cosmological parameters. It is essentially an error propagation process: it consistently propagates the uncertainty of the observables, to the errors on cosmological parameters.

The most relevant paper on Fisher matrix is [1], and in this lecture, we cover the key parts of it.

If $\mathbf{C}$ is the covariance matrix for a set of cosmological parameters θ, then,

$$(\mathbf{C}^{-1})_{ij} = \mathbf{F}_{ij} = \frac{\partial^2 \mathcal{L}}{\partial\theta_i \partial\theta_j} \tag{3}$$

where $\mathcal{L} \equiv \ln L$ and

$$L = \frac{\exp\left[-\frac{1}{2}(\boldsymbol{x}-\mu)^T \mathbf{\Sigma}^{-1}(\boldsymbol{x}-\mu)\right]}{\sqrt{(2\pi)^k |\mathbf{\Sigma}|}} \tag{4}$$

Dropping the 2π factors, we have,

$$2\mathcal{L} = \ln \det \mathbf{\Sigma} + (\boldsymbol{x}-\mu)^T \mathbf{\Sigma^{-1}} (\mathbf{x}-\mu) \tag{5}$$

Date: April 2, 2019.

Experiment	Observable	$\frac{1}{2}\mathrm{Tr}\left(\mathbf{\Sigma}^{-1}\mathbf{D}_i\mathbf{\Sigma}^{-1}\mathbf{D}_j\right)$	$\frac{\partial\mu}{\partial\theta_i}\mathbf{\Sigma}^{-1}\frac{\partial\mu}{\partial\theta_j}$
SNe	Luminosity distance	0	✓
BAO	$H(z), D_A(z)$	0	✓
RSD	$f\sigma_8(z)$	0	✓
CMB	C_ℓ; Angular power spectra	✓	0
Galaxy power spectra	$P(k)$	✓	0
Weak lensing	$C_\ell^{\gamma\gamma}$; Shear power spectra	✓	0

Let

$$\mathbf{\Sigma} = \left\langle (\boldsymbol{x}-\mu)^T(\boldsymbol{x}-\mu)\right\rangle; \quad \mathbf{D}_i = \frac{\partial\mathbf{\Sigma}}{\partial\theta_i}; \quad \mu = \langle\boldsymbol{x}\rangle. \tag{6}$$

Finally,

$$\mathbf{F}_{ij} = \frac{1}{2}\mathrm{Tr}\left(\mathbf{\Sigma}^{-1}\mathbf{D}_i\mathbf{\Sigma}^{-1}\mathbf{D}_j\right) + \frac{\partial\mu}{\partial\theta_i}\mathbf{\Sigma}^{-1}\frac{\partial\mu}{\partial\theta_j} \tag{7}$$

Note that there are two terms in the above general formula, but in practice, only one of these terms is nonzero, in many cases. Here's a list covering the observables we use in the data analysis.

References

[1] M. Tegmark, A. Taylor and A. Heavens, Astrophys. J. **480**, 22 (1997) doi:10.1086/303939 [astro-ph/9603021].

[2] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, "Numerical Recipes in FORTRAN: The Art of Scientific Computing."

Appendix A. The Fortran 90 code for SN Ia fitting

```
module precision
  integer, parameter :: dl = kind(1.d0)
end module precision

module cosmo
use precision
real(dl) :: om, ol, ok, H0=70.d0
real(dl) :: c =3.d5
real(dl) :: DH
real(dl), parameter :: om_min=0.d0, om_max=3.d0
real(dl), parameter :: ol_min=0.d0, ol_max=3.d0
integer,  parameter :: Np=100
```

```
end module cosmo


program test
use precision
use cosmo
implicit none

integer i, iz, iom, iol
integer, parameter :: n=580

real(dl)     :: z_dat(n), mu_dat(n), invcov(n,n)
real(dl)     :: diff(n), chi2
character    :: dummyc
real(dl)     :: dummyr

real(dl)     :: z

real(dl),external     :: E2, Einv, DC, DA, mu
logical :: flag=.false.

open(unit=50, file='sn_z_mu_dmu_plow_union2.1.txt')

do i=1, n
 read(50,*) dummyc, z_dat(i), mu_dat(i), dummyr, dummyr
 !write(*,*) z_dat(i), mu(i)
end do

close(50)

open(unit=50, file='sn_wmat_nosys_union2.1.txt')

do i=1, n
  read(50,*) invcov(i,:)
  !write(*,*) invcov(i,i)
end do

close(50)

open(unit=50, file='test_mu.dat')
```

```
do i=1, n
   write(50,*) z_dat(i), mu_dat(i), 1.d0/sqrt(invcov(i,i))
end do
close(50)

DH=c/H0

open(unit=50, file='chi2.dat')

do iom=1, Np

 om = om_min + (om_max-om_min)*dble(iom-1)/dble(Np-1)

do iol=1, Np

 ol = ol_min + (ol_max-ol_min)*dble(iol-1)/dble(Np-1)

 ok = 1.d0-om-ol

 call check(flag)

 if(flag) then

  do iz=1, n
    diff(iz)  = mu(z_dat(iz))-mu_dat(iz)
  end do ! loop for z

  chi2 = dot_product(diff,matmul(invcov,diff))

 else

  chi2 = 1.d10

 end if

 write(50,*) om, ol, chi2

 !om = om_min * (om_max/om_min)**(dble(i-1)/dble(Np-1))

end do ! loop for ol
```

```
end do ! loop for om


!z=0.5d0
!write(*,'(5e15.6)') z, E2(z), Einv(z), DC(z), DA(z), mu(z)

close(50)


end program test


subroutine check(flag)
use precision
use cosmo
implicit none

real(dl), parameter :: z_min=0.d0, z_max=3.d0
integer :: iz
integer, parameter :: nz=500
real(dl) :: z
real(dl), external :: E2, DA
logical :: flag

do iz=1, nz

 z = z_min +(z_max-z_min)*dble(iz-1)/dble(nz-1)

 if(E2(z)<0 .or. E2(z)==0 .or. DA(z)<0) then !! Thanks Yonghao
   flag = .false.
   return
 else
   flag = .true.
 end if

end do

end subroutine check
```

```
function E2(z)
use precision
use cosmo
implicit none

real(dl) :: z, E2

E2 = om*(1.d0+z)**3.d0+ol+(1.d0-om-ol)*(1+z)**2.d0


end function E2

function Einv(z)
use precision
use cosmo
implicit none

real(dl) :: z, Einv
real(dl), external :: E2

Einv = 1.d0/sqrt(E2(z))

end function Einv

function DC(z)
use precision
use cosmo
implicit none

real(dl), external :: rombint, Einv
real(dl), parameter :: tol=1.d-4
real(dl) :: DC, z

DC = rombint(Einv,0.d0,z,tol)
DC = DC*DH

end function DC

function DA(z)
use precision
```

```
use cosmo
implicit none

real(dl) :: DA, z, x
real(dl),external :: DC

if(ok>0) then

x= sqrt(ok)*DC(z)/DH

if(x<200.d0) then

DA = DH/(1.d0+z)/sqrt(ok)*sinh(sqrt(ok)*DC(z)/DH)

else

DA = 1.d0

end if

else if(ok<0) then

DA = DH/(1.d0+z)/sqrt(abs(ok))*sin(sqrt(abs(ok))*DC(z)/DH)

else

DA = DC(z)/(1.d0+z)

end if


end function DA


function Ldis(z)
use precision
use cosmo
implicit none

real(dl) :: Ldis, z
```

```
real(dl), external :: DA

Ldis = DA(z)*(1.d0+z)**2

end function Ldis


function mu(z)
use precision
use cosmo
implicit none

real(dl) :: mu, z
real(dl), external :: Ldis

 mu=5.d0*log10(Ldis(z))+25.d0

end function mu


       function rombint(f,a,b,tol)
        use Precision
!  Rombint returns the integral from a to b of using Romberg integration.
!  The method converges provided that f(x) is continuous in (a,b).
!  f must be real(dl) and must be declared external in the calling
!  routine.  tol indicates the desired relative accuracy in the integral.
!
        implicit none
        integer, parameter :: MAXITER=20
        integer, parameter :: MAXJ=5
        dimension g(MAXJ+1)
        real(dl) f
        external f
        real(dl) :: rombint
        real(dl), intent(in) :: a,b,tol
        integer :: nint, i, k, jmax, j
        real(dl) :: h, gmax, error, g, g0, g1, fourj
!

        h=0.5d0*(b-a)
```

```
      gmax=h*(f(a)+f(b))
      g(1)=gmax
      nint=1
      error=1.0d20
      i=0
10      i=i+1
        if (i.gt.MAXITER.or.(i.gt.5.and.abs(error).lt.tol)) &
          go to 40
!   Calculate next trapezoidal rule approximation to integral.
        g0=0._dl
          do 20 k=1,nint
          g0=g0+f(a+(k+k-1)*h)
20      continue
        g0=0.5d0*g(1)+h*g0
        h=0.5d0*h
        nint=nint+nint
        jmax=min(i,MAXJ)
        fourj=1._dl
          do 30 j=1,jmax
!   Use Richardson extrapolation.
          fourj=4._dl*fourj
          g1=g0+(g0-g(j))/(fourj-1._dl)
          g(j)=g0
          g0=g1
30      continue
        if (abs(g0).gt.tol) then
          error=1._dl-gmax/g0
        else
          error=gmax
        end if
        gmax=g0
        g(jmax+1)=g0
      go to 10
40    rombint=g0
      if (i.gt.MAXITER.and.abs(error).gt.tol)  then
        write(*,*) 'Warning: Rombint failed to converge; '
        write (*,*)'integral, error, tol:', rombint,error, tol
      end if

      end function rombint
```