LECTURE 5: FUNDAMENTALS OF OBSERVATIONAL COSMOLOGY (I)

GONG-BO ZHAO

In this lecture, we start covering basics of observational cosmology from analyzing the type Ia supernovae (SN Ia) data. As we discussed in the lecture, SN Ia are 'standard candles' of the Universe, which can be used to measure the luminosity distances. On the other hand, the redshifts of SN Ia can be independently measured by taking the spectra. Combining redshifts and distances, one can infer cosmological parameters. This is exactly what Adam Riess and Saul Perlmutter did in their Nobel-winning papers in 1998 [1, 2] (these papers have been cited over 10,000 times each)!

So the aim of this lecture and the next one is to reproduce the key results in the above-mentioned famous papers, *i.e.*, discover the cosmic acceleration using the SN Ia data!

The observed dataset of SN Ia is actually a list of redshifts z_i, the distance module μ_i, and a data covariance matrix $\mathbf{C}$. Note that,

$$\mu(z) \equiv m - M = 5\,\log_{10} D_L(z) + 25 \tag{1}$$

$$D_L = \begin{cases} (1+z)\frac{D_H}{\sqrt{\Omega_K}}\,\sinh\,\left(\sqrt{\Omega_K}\frac{D_C}{D_H}\right) & \Omega_K > 0; \\ (1+z)D_C & \Omega_K = 0; \\ (1+z)\frac{D_H}{\sqrt{-\Omega_K}}\,\sin\,\left(\sqrt{-\Omega_K}\frac{D_C}{D_H}\right) & \Omega_K < 0. \end{cases} \tag{2}$$

$$D_C = D_H \int_0^z \frac{\mathrm{d}z'}{E(z')} \tag{3}$$

$$E(z) = \sqrt{\Omega_\mathrm{M}(1+z)^3 + \Omega_\Lambda + (1-\Omega_\mathrm{M}-\Omega_\Lambda)(1+z)^2} \tag{4}$$

$$D_H = \frac{c}{H_0} \tag{5}$$

These equations are sufficient to allow for evaluating μ at arbitrary redshifts, given a set of $\{\Omega_\mathrm{M}, \Omega_\Lambda, H0\}$.

In class, we have written a Fortran code for this purpose, which is included in the Appendix. Note that the subroutine to perform the integral, `rombint`, is taken from Numerical Recipe [3], which is available online, `http://www.nrbook.com/a/bookfpdf.php`

References

[1] A. G. Riess *et al.* [Supernova Search Team], "Observational evidence from supernovae for an accelerating universe and a cosmological constant," Astron. J. **116**, 1009 (1998) doi:10.1086/300499 [astro-ph/9805201].

[2] S. Perlmutter *et al.* [Supernova Cosmology Project Collaboration], "Measurements of Omega and Lambda from 42 high redshift supernovae," Astrophys. J. **517**, 565 (1999) doi:10.1086/307221 [astro-ph/9812133].

[3] 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 computing the luminosity distance

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

module cosmo
use precision
real(dl) :: om=0.3d0, ol=0.7d0, ok, H0=70.d0
real(dl) :: c =3.d5
real(dl) :: DH
end module cosmo


program test
use precision
use cosmo
implicit none

integer i
integer, parameter :: n=580

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

real(dl)      :: z

real(dl),external     :: E2, Einv, DC, DA, mu
```

```
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)

ok = 1.d0-om-ol

DH=c/H0

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

end program test



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

```fortran
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
real(dl),external :: DC
```

```
if(ok>0) then

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

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
```