Users' Guide for CSWAB 2.0

Hu Zhan
Department of Physics, University of California, Davis, CA 95616, USA
zhan@physics.ucdavis.edu

January 26, 2008

Contents

1 Introduction

CSWAB utilizes the Fisher information matrix to forecast constraints on cosmological (and other) parameters from the **C**osmic microwave background (CMB), type Ia **S**upernova (SN), **W**eak lensing (WL), **A**nd **B**aryon acoustic oscillation (BAO) data. This version covers photometric WL, SN, and BAO analyses as well as spectroscopic SN and 3-dimensional BAO analyses.

The calculations involved are described in Zhan (2006), Zhan & Knox (2006a), and Zhan et al. (2008). This code is primarily written in `C++` in a quasi-object-oriented fashion, so that it may be easily adapted for other applications. CSWAB 2.0 contains 5 packages: auxiliary functions, evolution of cosmological quantities (including cosmological parameters, distance, linear growth function, etc.), CMB, Angular WL/BAO and three-dimensional BAO, and SN with each one depending on the preceding one(s).

CSWAB takes advantage of the GNU Scientific Library[1] (GSL, version 1.6) and LAPACK[2] for numerical algorithms and relies on CMBFAST version 4.5.1[3] (Zaldarriaga & Seljak 2000) for CMB power spectra and transfer functions. It is parallelized through the massage passing interface[4] (MPI, standard 2.0). Although one has the option to compile and run CSWAB as a serial application, it may take too long (e.g., days) and/or too much memory to carry out a production run of, say, 50 galaxy and shear bins on a single computer.

Cosmological parameters forecast in CSWAB 2.0 include the dark energy equation of state (EOS) parameters w_0 and w_a, where the EOS is parametrized as $w(a) = w_0 + w_\mathrm{a}(1-a)$, the matter density ω_m, the baryon density ω_b, the angular size of the sound horizon at the last scattering surface θ_s, the equivalent matter fraction of curvature Ω_K, the optical depth τ to scattering by electrons in the reionized intergalactic medium, the primordial helium mass fraction Y_p, the spectral index n_s of the primordial scalar perturbation power spectrum, the running of the spectral index α_s, and the normalization of the primordial curvature power spectrum Δ^2_R at $k^* = 0.05\,\mathrm{Mpc}^{-1}$. I adopt the three-year *WMAP* results Spergel et al. (2007) for fiducial values of the parameters: (w_0, w_a, ω_m, ω_b, θ_s, Ω_K, τ, Y_p, n_s, α_s, Δ^2_R) = (-1, 0, 0.127, 0.0223, 0.596°, 0, 0.09, 0.24, 0.951, 0, 2.0×10^{-9}). The reduced Hubble constant $h = 0.73$ and the present equivalent matter fraction of dark energy $\Omega_\mathrm{X} = 0.76$ are implicit in my parametrization.

I assume in this guide that you have unpacked `cswab2.tgz` and that the current directory is the CSWAB home directory: `some_directory/cswab2/`. File names, directory names, commands, and file contents are in typewriter font. Optional arguments are enclosed in square brackets.

Bugs/Questions: Email zhan@physics.ucdavis.edu.

Disclaimer: CSWAB is provided as is and without warranties of any kind. The user must agree to use CSWAB at his/her own risk.

2 Incremental Changes w.r.t. Version 1.0

2.1 Bugs Fixed

I. The running, α_s, of the primordial power spectral index. By definition,

$$n_\mathrm{s} - 1 = \left.\frac{\partial \ln \Delta^2_\mathrm{R}(k)}{\partial \ln k}\right|_{k=k^*} \quad \text{and} \quad \alpha_\mathrm{s} = \left.\frac{\partial^2 \ln \Delta^2_\mathrm{R}(k)}{\partial \ln^2 k}\right|_{k=k^*} = \left.\frac{\partial n_\mathrm{s}}{\partial \ln k}\right|_{k=k^*}.$$

Hence, $\Delta^2_\mathrm{R}(k) = \Delta^2_\mathrm{R}(k^*)(k/k^*)^{n_\mathrm{s}-1+0.5\alpha_\mathrm{s}\ln(k/k^*)}$ is accurate to second order *for the power spectrum*, and $n_\mathrm{s}(k) = n_\mathrm{s}(k^*) + \alpha_\mathrm{s}\ln(k/k^*)$ is accurate to first order *for the spectral index.*

[1]See http://www.gnu.org/software/gsl/.
[2]See http://www.netlib.org/lapack/.
[3]See http://www.cmbfast.org/.
[4]See http://www-unix.mcs.anl.gov/mpi/.

Although I initiated the matter power spectrum from CMBFAST with the correct factor of 1/2 in front of α_s, it was not present in the no-wiggle power spectrum (for nonlinear evolution, see § 3). Restoring this factor has made the error on α_s much larger but no significant impact on other parameters.

Separately, version 1.0 extends the matter power spectrum beyond the range of $10^{-4} \leq k \leq 10\,h^{-1}$Mpc by power law with a fixed index at either end. Current version includes the effect of α_s, though I am sure that it is still not correct on the small-scale end. In any case, there is not much impact on the forecasts.

II. Derivatives w.r.t. Δ^2_R. In version 1.0, Δ^2_R is merely a multiplicative factor to the matter power spectrum. However, the amplitude of the matter power spectrum affects nonlinear evolution, so more information could be extracted. Allowing Δ^2_R to affect the nonlinear power spectrum improves the constraint on Δ^2_R. It is arguable whether we will have sufficient knowledge about the nonlinear power spectrum by the time the data are available, but the impact of this change on other parameters is small.

2.2 New Components and Tools

I. Two-stage three-dimensional BAO forecasts (Seo & Eisenstein 2003; Zhan & Knox 2006a) with intermediate results on distance, Hubble parameter, and other parameters. This is set by the compiling option `-DBAO3DDH`.

II. Direct three-dimensional BAO forecasts on cosmological and other parameters (Zhan & Knox 2006a). The direct results should be similar to the final two-stage results above.

III. Spectroscopic and photometric SN cosmology (Knox, Song, & Zhan 2006; Zhan et al. 2008).

IV. A handy utility `zcos` for calculating comoving distance $D(z)$, growth rate $G(z)$, growth index $f(z) = d\ln G/d\ln a$, Hubble parameter $H(z)$, and time $t(z)$ for a given cosmological model and the inverse $z(D)$, $z(G)$, $z(H)$, and $z(t)$.

V. Adaptive cubic spline interpolation. The class `AdaptiveInterp` in `src/auxi/adpintp.h` assigns more points where largest interpolation errors would occur at uniform intervals to ensure the precision and efficiency of the interpolation. It replaces the fixed-interval interpolations in the previous version.

2.3 Other Changes

I. CSWAB 2.0 has migrated to MPI standard 2.0, which provides one-sided communications and process spawning. I take advantage of the new features to dynamically balance the load on each computing node and significantly improve the efficiency. This move, however, requires you to tweak your own copy of MPICH2[5] and put up with the somewhat awkward MPD-Ring mechanism. See § 4.2 for more details.

I will no longer actively maintain the serial version but will provide support if problems are reported. You can always install MPICH2 on a single-processor computer and run an application, e.g., `abcd`, with `mpiexec -n 1 abcd`... If you always run CSWAB 2.0 on a single processor, you do not need to tweak MPICH2.

II. The unit of distance in `src/gals` has been switched from h^{-1}Mpc to Mpc (h-less), so k is in Mpc^{-1} and the power spectrum $P(k)$ in Mpc^3. The rationale for this change is that the

[5]See http://www.mcs.anl.gov/research/projects/mpich2/.

transfer function $T(k)$ does not depend on w_0, w_a, θ_s, or Ω_K if k is in Mpc^{-1}. CMBFAST does produce different $T(k)$ that are merely shifted with respect to each other. That is because CMBFAST outputs k in $h\,\mathrm{Mpc}^{-1}$ and includes the growth rate in $T(k)$.

Note that the `src/cos` package remains h-full.

III. The class `BAOWL` in `src/gals/fisher.h` no longer calculates the Fisher matrix; it only writes the power spectra to files. The Fisher matrix calculation is now performed solely by the class `CosmicTomography`.

IV. The power spectra are sorted according to their ℓ_max in the output files. This means that version 1.0 outputs are not compatible with this version.

V. Multiplicative shear errors are introduced, and the additive shear errors now have more freedom:

$$(\boldsymbol{C}_\ell)_{ij}^{\gamma\gamma} = (1 + f_i + f_j)P_{ij}^{\gamma\gamma}(\ell) + \delta_{ij}^\mathrm{K}\gamma_\mathrm{rms}^2/\bar{n}_i + A_i^\gamma A_j^\gamma.$$

The fiducial value of f_i is taken to be 0. The additive errors here are still not as flexible as those in Huterer et al. (2006), but it is the amplitude of A_i^γ that matters the most. Similarly, galaxy power spectra have the additive errors A_i^g, which are assumed to be uncorrelated with A_i^γ. The multiplicative errors for the galaxy power spectra are absorbed in the galaxy bias parameters.

3 Limitation

CSWAB 2.0 is based on very ideal assumptions. For example, I have not considered non-Gaussian photo-z errors, or different galaxy bias and photo-z properties associated with galaxy types (e.g., Schneider et al. 2006), or more complex behavior of the systematics (e.g., Huterer et al. 2006).

I should also mention that the transformation between the linear matter power spectrum to the nonlinear one in CSWAB is *ad hoc*. Although the Peacock & Dodds (1996) fitting formula works for a no-wiggle power spectrum, it does not make sense for a "real" power spectrum, in which BAOs are present. The reasons are that it shifts the scales of BAO features unphysically:

$$k_\mathrm{L} = [1 + \Delta_\mathrm{NL}^2(k_\mathrm{NL})]^{-1/3}k_\mathrm{NL}, \tag{1}$$

$$\Delta_\mathrm{NL}^2(k_\mathrm{NL}) = f_\mathrm{NL}[\Delta_\mathrm{L}^2(k_\mathrm{L})], \tag{2}$$

where $f_\mathrm{NL}(x)$ is determined by the spectral index $n_\mathrm{L}(k_\mathrm{L}) = \mathrm{d}\ln P/\mathrm{d}\ln k|_{k=k_\mathrm{L}/2}$ (and the growth function), and that the mapping can be non-monotonic due to the oscillations in n_L. Other fitting formulae that depend on the spectral index may face the same problem. My solution is to use a no-wiggle power spectrum $\Delta_\mathrm{NW,L}^2(k)$ (Eisenstein & Hu 1999) that otherwise matches the real power spectrum $\Delta_\mathrm{L}^2(k)$ with BAOs to obtain the boost $\Delta_\mathrm{NW,NL}^2(k)/\Delta_\mathrm{NW,L}^2(k)$ and then apply it to $\Delta_\mathrm{L}^2(k)$.

Alternatively, one may obtain the nonlinear power spectrum with BAOs using the halo model (e.g., Cooray & Sheth 2002), perturbation theory (Jain & Bertschinger 1994; Jeong & Komatsu 2006), and simulations (Seo & Eisenstein 2005; Springel et al. 2005; White 2005). These methods all have their own limitations: the halo model is less accurate in the quasi-linear regime (the computation is more intensive as well), perturbative calculations diverge at $z \lesssim 1$, and simulations are time consuming (ultimately, the nonlinear power spectrum should be calibrated by simulations).

Given my treatment of the nonlinear power spectrum and the fact that I do not include neutrinos, results of the parameters related to the power spectral shape, such as the initial tilt n_s and running α_s, may be less robust.

4 Building CSWAB

Before compiling CSWAB, you need to set up the dependencies: GSL, LAPACK, MPICH2, and CMBFAST version 4.5.1. Follow standard procedures to install GSL and LAPACK. Specific modifications to CMBFAST and MPICH2 and instructions for CSWAB are described as follows.

4.1 Interfacing with CMBFAST

One powerful CMB constraint is the angular size of the sound horizon at the last scattering surface, $\theta_{\rm s}(w_0, w_{\rm a}, \omega_{\rm m}, \omega_{\rm b}, Y_{\rm p}, \Omega_{\rm K}, \Omega_{\rm X})$. It is a useful and important parameter to include for forecasting CMB constraints or incorporating CMB priors with forecasts of other experiments.

In order to take numerical derivatives of the power spectra with respect to other parameters at a fixed $\theta_{\rm s}$, one must trade the implicit parameter $\Omega_{\rm X}$ with the others. Fitting formulae are convenient for approximating $\theta_{\rm s}$ to the percent level, but given that $\theta_{\rm s}$ can be so precisely measured, e.g., $\sigma(\theta_{\rm s}) \sim 0.0002^\circ$ from *Planck*, these formulae may not be accurate enough for determining the exact trade-offs between $\Omega_{\rm X}$ and the other parameters.

To build a consistent forecasting tool, I extract $\theta_{\rm s}$ from CMBFAST, which supplies the distance to the peak of the visibility function (occurred at `armax`), `dlsso`, and the sound horizon at the peak, `shor`, in `subroutine cmbflat` (`cmbflat.F`) and `subroutine cmbopen` (`cmbopen.F`). Hereafter, I only cite the flat case for brevity. A naive extraction of $\theta_{\rm s}$ may lead to errors as large as $\Delta\theta_{\rm s} \sim 0.0001^\circ$. *This does not affect CMBFAST results*, because `dlsso`, `shor`, and `armax` are not used in any way that can impact the computation in CMBFAST appreciably. Rather, the errors will affect the calculations of the input parameters of CMBFAST and lead to incorrect numerical derivatives.

The error of the extracted $\theta_{\rm s}$ is primarily caused by the fact that the extremum is searched over arrays in `subroutine finithermo` (`cmbflat.F`, line 3067) without interpolation. Other factors exacerbate the problem, but their own contribution to the error – when `armax` is interpolated – is $\lesssim 10^{-5}$ deg. For instance, one can get as much as $\sim 0.0001^\circ$ differences in $\theta_{\rm s}$ by assigning different values to the maximum wavenumbers `akmaxt` for transfer functions or `akmax0` for CMB. The reason is that both `akmaxt` and `akmax0` affect the starting point, `taumin`, for initializing the thermal/ionization history of baryons in `subroutine finithermo` and, hence, the peak can shift from one place in the arrays to another. Interested readers may want to check the value of `shor/dlsso` in the original CMBFAST as `akmaxt` varies (try `cmbfast_directory/cmb < src/cmb/k1.par` and `cmbfast_directory/cmb < src/cmb/k100.par`).

By the same token, the specified fiducial value of $\theta_{\rm s}$ is not precise enough. Therefore, the actual fiducial value is calculated from the other parameters.

I list the essential modifications in this subsection; for others, see the results of `diff` between the modified and original CMBFAST codes in `src/cmb/cmbfast.dif`. After finishing the modifications, compile CMBFAST with `HP` and `UNNORM` options. Do not use `MPIBUILD`.

4.1.1 Increasing the Resolution in `cmbfast.inc` and `subroutines.F`

Modifications are made to increase the resolution in general. For `cmbfast.inc`:

I. *Position* Line 29
Original `parameter(nthermo=10000)`
Modified `parameter(nthermo=40000)`
Remarks To increase `nthermo` above 40000, one must increase the array sizes in `subroutine splder` and `subroutine splini`, which are called by `subroutine finithermo`.

II. *Position* Line 38
Original `parameter (nk0=300)`

Modified `parameter (nk0=1200)`
Remarks To get smooth BAOs, `src/cmb/cmb.h` sets the output frequency, `NK_PER_DEX`, of the transfer function to 150 per dex of wavenumber. CMBFAST starts output at $k \sim 10^{-5}\ h\,\mathrm{Mpc}^{-1}$. To get $k \sim 10\ h\,\mathrm{Mpc}^{-1}$ for WL, you need `nk0` $\gtrsim 900$.

III. *Position* Line 40
Original `parameter (nstep0=7000)`
Modified `parameter (nstep0=15000)`
Remarks Required for the concordance model and small perturbations around it.

Modifications to `subroutines.F`:

I. *Position* Line 1023 & 1025, `subroutine output_power`
Original `parameter(nmax=1000)`
`parameter(kmax=1000)`
Modified `parameter(nmax=nk0)`
`parameter(kmax=nk0)`
Remarks To avoid indexing beyond the array bound.

4.1.2 Extracting the Sound Horizon from `cmbflat.F` and `cmbopen.F`

The following code fragment from `cmbflat.F` (line 363)

```
call finithermo(taumin,taumax,tau0,taurend,dlntau0,n1,nstep)
dlsso=tau0-taurmax
shor=rombint(dsoundda,1.0d-8,armax,tol)
```

gives the distance to the peak of the visibility function (occurred at the conformal time `taurmax` or the expansion factor `armax`), `dlsso`, and the sound horizon at `armax`, `shor`. As mentioned above, the extremum is searched over arrays without interpolation in `subroutine finithermo` (line 3067):

```
vismax=0.0d0
do i=1,nthermo
   tvis=dotmu(i)*emmu(i)
   if (tvis.gt.vismax) then
      vismax=tvis
      tau=tauminn*exp((i-1)*dlntau)
      taurmax=tau
      armax=atemp(i)
   endif
end do
```

Consequently, the numerical derivatives can be erratic. To alleviate this problem, I supplement the search with an interpolation to the point where the first derivative of the visibility function vanishes:

```
vismax=0.0d0
do i=1,nthermo
   tvis=dotmu(i)*emmu(i)
   if (tvis.gt.vismax) then
      vismax=tvis
      tau=tauminn*exp((i-1)*dlntau)
      taurmax=tau
      armax=atemp(i)
      nvmx1 = i
```

```
         endif
      end do

      dv1 = ddotmu(nvmx1) * emmu(nvmx1) + dotmu(nvmx1) * demmu(nvmx1)
      if (dv1 .ge. 0.d0) then
         nvmx2 = nvmx1 + 1
      else
         nvmx2 = nvmx1 - 1
      endif
      dv2 = ddotmu(nvmx2) * emmu(nvmx2) + dotmu(nvmx2) * demmu(nvmx2)
      i = nvmx2 - 1
      taurmax = (tau * dv2 - tauminn * exp(i * dlntau) * dv1)
     &      / (dv2 - dv1)
      armax = (armax * dv2 - atemp(nvmx2) * dv1) / (dv2 - dv1)
```

Now θ_s responds more smoothly to the parameters. Similar changes to `cmbopen.F` are documented in `src/cmb/cmbfast.dif`.

4.1.3 Curvature in `recfast.f`

Finally, there is a minor bug in `subroutine recfast(OmegaB,OmegaC,omegav,H0inp,tcmb,Yp, annur)` (`recfast.f`, line 256):

```
      OmegaT=OmegaC+OmegaB            !total dark matter + baryons
      OmegaK=1.d0-OmegaT-OmegaL       !curvature
      Omegal=omegav
```

where the last two lines should have been switched.

Since `OmegaL` (case insensitive) is never assigned before being used, the curvature term `OmegaK` is indeterminant. Fortran compilers may set all automatic variables to 0 *once* when the program starts (*not* every time the subroutine is called), so that one gets `OmegaK=1.d0-OmegaT` regardless the input value of `omegav`. If the subroutine is called a second time, then `OmegaL` may retain its value from the first call until being reassigned (again, depending on the behavior of the compiler), which could result in a correct value of `OmegaK` if there were no changes in `OmegaB`, `OmegaC`, and `omegav` between the two calls. The impact of this bug *alone* on θ_s is $\lesssim 10^{-5}$ deg, because `OmegaL` ~ 0 to high degree at recombination.

4.2 Tweaking MPICH2

MPI 2.0's one-sided communications and process spawning can be used to distribute the jobs efficiently at run time. I let one computing node spawn an idling process, which hosts a job queue for other processes to access and modify. Unfortunately, MPI 2.0 standard neither provide a bundled fetch-and-modify operation, nor does it guarantee the order in which the one-sided communications are completed, so I (you) have to tweak the MPICH2 code. The good news is that you do not need root privilege to install and use a copy of MPICH2 in your home directory. I have put all the relevant code in `src/auxi/zunlock.cpp`. What you have to do is to append that file to MPICH2 source file `mpich.../src/mpid/ch3/src/ch3u_rma_sync.c`, compile MPICH2, and set up the paths to headers, libraries, and executables correctly.

A cumbersome aspect of MPICH2 is that it runs over an MPD-Ring. You have to ensure that the ring persists over the computing nodes during the lifetime of an MPI 2.0 job. See MPICH2 User's Guide for more details. The SGE queue script `example/cmb/cmb.que` shows some tricks to work between the queue management and MPD-Ring. My experience is that two rings cannot have overlapping computing nodes, even if they do not overlap in terms of processors. Otherwise, the second ring will break the node from the first ring, and your first job will hang. This means that you can only run one MPICH2 job at a time on a cluster or that the queue management must be configured not to overlap jobs on the same node, which is not likely given the popularity of multi-core nodes. One solution is to establish a ring on all nodes of a cluster and let all your jobs use the existing ring without starting new ones. However, this is inappropriate on a large cluster. Let me know if you have a solution that does not require administrative privilege.

4.3 Building CSWAB: A "Quick" Start

First, you need to set several path names in `src/Makefile`:

```
JOBDIR  = directory_of_cswab2
CMBFDIR = directory_of_cmbfast
SYSINCDIR = directory_of_GSL_LAPACK_and_MPI_headers
SYSLIBDIR = directory_of_GSL_LAPACK_and_MPI_libraries
```

The last two variables may be left empty if GSL, LAPACK, and MPICH2 can be found in the default system header and library directories. Then, choose `C++` and Fortran compilers and set their switches:

```
CXX = mpicxx
CXXFLAGS = -O3 -Wall -Wno-deprecated
FC  = g77
FFLAGS = -O3 -DHP -DUNNORM -I$(CMBFDIR)
```

Here `mpicxx` is an MPI 2.0 wrapper of `g++` on my system. Finally, set the MPI option and variables (choose a parallel `C++` compiler above as well) if you want a parallel CSWAB:

```
CXXFLAGS += -DMPI_CODE -DMPICH_IGNORE_CXX_SEEK
CXXFLAGS += -DTWEAK_MPI_RMA_SYNC -DTWEAK_MPI_DEBUG
OPTIONS += -DJOBCNTL=\"$(JOBDIR)/bin/jobcntl\"
MPIRUN = /usr/bin/mpiexec -n 2
MPILIB = mpich
```

Otherwise, these entries should be left empty.

Now, type `make -C src`. If everything goes well, seven executables: `cmbfm`, `snefm`, `wlbao`, `clcon`, `bao`, `zcos`, and `jobcntl` will be generated in `bin/` in a short while.

4.4 Examples

Pre-calculated examples are available for CMB and WL/BAO based on *Planck* and the Large Synoptic Survey Telescope[6] (LSST), respectively. You need to download the CMB power spectra, transfer functions, and shear and galaxy power spectra (54MB gzipped) from `software.hzhan.net`. Three-dimensional BAO and SNe can be calculated on the fly. Once the files are in place (`example/`), you can run them by issuing the command `make -C src example`. The results should be approximately the same as those in `example/log`. Detailed output log of each component can be found in the sub-directories in `example/`.

To run full tests, issue the command `make -C src fulltest`. It takes about 3 hours to finish on a dual Opteron 1.8 GHz. Note that this will overwrite the existing examples, so *make a copy of them before running the full tests*. Once the power spectra are calculated, just `make -C src example` to check the results.

[6]See http://www.lsst.org/.

4.5 Other Options

You may turn on a number of options in `src/Makefile` with `OPTIONS += -Doption_name`[=*value*]. These options are grouped in the following categories.

I. CMB experiment: `CMBPOL`, `PLANCK`, & `WMAP`

   Choose *one* of the options for the noise characteristics of the CMB experiment, which are hardwired in `src/cmb/cl.cpp`. You have to specify the correct sky fraction at the command line and, if desired, modify the ℓ ranges in `src/cmb/cmb.h`.

II. Debugging: `VERBOSE`, `OUT_CL`, & `REMOVE_OLD_CLS`

   Setting `VERBOSE=0` (or not switching it on at all) leads to no extra output; `1`, `2`, and `3` produce progressively more verbose logs. Set `OUT_CL=`n, where n is an integer, to output $P_{ij}(\ell)$ and the derivatives at $\ell = n$. If `REMOVE_OLD_CLS` is set, existing shear and galaxy power spectra and bin information will be deleted; if not, they will be renamed with a suffix of `bbk` or `tbk`, depending on whether the file is binary or text.

III. Power Spectrum: `LINEAR_PS`, `NINJCL`, & `SHIFT_PS`

   Turn on `LINEAR_PS` to compute the galaxy and shear power spectra using the linear matter power spectrum. `NINJCL` enables one to calculate the Fisher matrix with $\bar{n}_i\bar{n}_j P_{ij}(\ell)$ instead of $P_{ij}(\ell)$ (Ma, Hu, & Huterer 2006). Note that `clcon` and `wlbao` must be compiled with the same `NINJCL` choice (set or not) and that if `NINJCL` is set, all the shear and galaxy power spectra files are written with the $\bar{n}_i\bar{n}_j$ factors ($\bar{n}_i$ in steradian^{-1}). `SHIFT_PS` shifts the amplitude of the matter power spectrum for perturbations in w_0, w_a, θ_s, and Ω_K as discussed in § 2.3; if not set, the power spectra will be read from CMBFAST outputs.

IV. Growth for WL: `WL_NO_G`

   This option is devised to investigate the origin of cosmological constraints from WL. `WL_NO_G` assigns artificial bias parameters [fiducial value $b(z) = 1$] to WL shear power spectra (Knox, Song, & Zhan 2006; Zhan & Knox 2006b). It not only limits WL's ability to measure the amplitude (or growth rate) of the matter power spectrum, but also degrades WL's distance measurements.

V. Galaxy bias: `GB_SPL_INTP` & `GALB_DLNZ1`

   The galaxy bias is linearly interpolated from the bias parameters. If you want to use cubic spline interpolation, set the option `GB_SPL_INTP`. The spacing of the galaxy bias parameters is the same as that of the photo-z parameters: uniform or proportional to $(1+z)$ from $z = 0$ to `Z_MAX` + `Z_EXT` (§ 4.6), as set in the WL/BAO parameter file (§ 5.3). Turn on the option `GALB_DLNZ1` if you want to assign the galaxy bias parameters at even intervals in $\log(1+z)$ from $z = 0$ to `Z_NO_GAL` (§ 5.3) beyond which no galaxy is present in the survey for our purpose. `GALB_DLNZ1` will also cause the interpolation to be based on $b(z_i)$ vs. $\log(1+z_i)$ as opposed to $b(z_i)$ vs. z_i.

VI. Three-dimensional BAO: `BAO3DDH`

   Set the option `BAO3DDH`, if you want a two-stage three-dimensional BAO forecast. Note that you must instruct `bin/bao` to read the appropriate transfer functions that are generated for this purpose (§ 5.1 & 5.4).

4.6 Obsolete Options

The two previously recommended options `DA_F_DB` and `EXTEND_PARS` are now enforced and, hence, are obsolete.

I. Fisher Matrix Algorithm: `DA_F_DB`

For zero-mean observables (e.g., the map data in Fourier space) with no scale-scale correlations, one may calculate the Fisher matrix in two ways (Hu & Jain 2004):

$$F_{\alpha\beta} = f_{\rm sky} \sum_\ell \frac{2\ell+1}{2} \,\mathrm{Tr}\, \boldsymbol{C}_\ell^{-1} \frac{\partial \boldsymbol{C}_\ell}{\partial p_\alpha} \boldsymbol{C}_\ell^{-1} \frac{\partial \boldsymbol{C}_\ell}{\partial p_\beta} \tag{3}$$

$$= f_{\rm sky} \sum_\ell (2\ell+1) \frac{\partial \boldsymbol{Q}_\ell^{\rm T}}{\partial p_\alpha} \boldsymbol{M}_\ell^{-1} \frac{\partial \boldsymbol{Q}_\ell}{\partial p_\beta}, \tag{4}$$

where p_α is a parameter, $\boldsymbol{C}_\ell$ is the covariance, i.e., the auto and cross power spectra, of the observables in Fourier space, $\boldsymbol{Q}_\ell$ is a column vector of unique elements of $\boldsymbol{C}_\ell$, and $\boldsymbol{M}_\ell$ is the covariance of $\boldsymbol{Q}_\ell$. $\boldsymbol{M}_\ell$ can be constructed analytically with the elements of $\boldsymbol{C}_\ell$.

Equation (3) is easy to implement but inefficient and sometimes unstable. For example, although $\boldsymbol{C}_\ell^{-1}$ is supposed to be symmetric, transposing it once in the equation (or just taking "advantage" of the symmetric matrix product in `BLAS`) may lead to large errors. It is recommended to polish the inverse $\boldsymbol{C}_\ell^{-1}$ a few times for equation (3).

II. Redshift Range of Parameters: `EXTEND_PARS`

The galaxy bias and photo-z parameters are assigned in true-redshift space; they have nothing to do with the binning of galaxies in photo-z space. Without `EXTEND_PARS`, the parameters are spaced between $z = 0$ to (the numerical value of) the maximum photo-z $z_{\rm p,max} =$ `Z_MAX` (`src/gals/swab.h` & `swab.cpp`) of the survey. Because of photo-z errors, the galaxy distribution in true-redshift space spreads beyond $z =$ `Z_MAX`. Therefore, it is reasonable to extend the parameters to $z =$ `Z_MAX` + `Z_EXT` with the option `EXTEND_PARS`, where `Z_EXT` = 0.5 can be adjusted in `src/gals/swab.cpp`. This extension does not add more parameters but only increase the redshift interval between parameters.

5 Running CSWAB

There are six driver programs `cmbfm.cpp`, `snefm.cpp`, `wlbao.cpp`, `clcon.cpp`, and `bao.cpp` in `src/`, and `zcos.cpp` in `src/cos/`. They provide simple examples of utilizing CSWAB classes and are flexible enough for a variety of applications. The calculations are described in Zhan (2006), (Zhan & Knox 2006a), and (Zhan et al. 2008). For brevity, I define a Boolean argument `x` to be *true* if and only if `x` can be interpreted as a non-zero integer or a string starting with `t` or `y` (case insensitive).

5.1 CMB

Program `bin/cmbfm`

Purpose To generate the CMB power spectra, transfer functions, and CMB Fisher matrix.

Syntax `cmbfm f_sky Cl_dir cmb_FM [cal_Cls cos_par cmbfast for_DGH]`

`f_sky`	Effective sky fraction covered by the experiment.
`Cl_dir`	Directory to write and/or read CMB power spectra and transfer functions.
`cmb_FM`	File name of the output Fisher matrix .

`cal_Cls`	Calculate the CMB power spectra and transfer functions if `cal_Cls` is *true* (default *false*).
`cos_par`	Cosmological parameter file. If not specified, the program assumes the concordance model in `src/cmb/cmb.cpp`.
`cmbfast`	CMBFAST executable name (default `CMBF_EXEC` in `src/Makefile`).
`for_DGH`	Calculate the transfer functions for two-stage BAO if `for_DGH` is *true* (default *false*).

Remarks Pre-calculated CMB power spectra must exist in the directory `Cl_dir` if `cal_Cls` is *false*. Otherwise, `Cl_dir` must be writable.

Here is an excerpt from the cosmological parameter file `example/cmb/lcdm.par`:

```
% comments start with %, #, or !
w0  -1  2e-2  2
wa   0  4e-2  2
...
```

The columns are the parameter name (string), its fiducial value (float), the step size for numerical differentiation (float), and the method of differentiation (integer, always two-sided). The parameter names are stored in `COS_PAR_NAME` (`src/cmb.cpp`). The order of the parameters is not important. The fiducial value of θ_s will be calculated from other parameters, so that you need to specify either `Ox` for Ω_X or `H0` for H_0. Only the fiducial value is relevant to `Ox` and `H0`. If both are given, `Ox` overrides `H0`. Entries that do not match the defined names will be discarded. If multiple entries of the same parameter are present, the last one will be in effect. Parameters that are absent in the file will assume the values in `src/cmb/cmb.cpp`.

The order of parameters in the CMB Fisher matrix follows: w_0, w_a, ω_m, ω_b, reserved, θ_s, Ω_K, τ, Y_p, n_s, α_s, reserved, and Δ^2_R.

5.2 Supernovae

Program `bin/snefm`

Purpose To generate a SN Fisher matrix.

Syntax `snefm cos_par z_beg z_end sig_z0 n_phz n_sne dndz sig_m [n_f sig_f sne_FM cal_FM pri_evo pri_dz cmb_FM pri_lnh]`

`cos_par`	Cosmological parameter file. It must be the same as that used by `bin/cmbfm`.
`z_beg`	Beginning redshift of the SN distribution. For photo-z SN data, `z_beg` is the beginning photo-z.
`z_end`	End redshift of the SN distribution. For photo-z SN data, `z_end` is the end photo-z.
`sig_z0`	Photo-z rms error at $z = 0$, σ_{z0}, as in $\sigma_z = \sigma_{z0}(1+z)$. I assume that there is no photo-z bias, because a known bias has little impact on the results; it is the uncertainty in the bias that matters. For spectroscopic SN data, run `bin/snefm` with `sig_z0` $= 0$.
`n_phz`	Number of SN photo-z parameters in each component, i.e., `n_phz` photo-z rms and `n_phz` photo-z bias parameters. Since the photo-z rms and bias are linearly interpolated from the parameters, `n_phz` ≥ 2. For spectroscopic SN data, set `n_phz` $= 0$ (although the program will still initialize a `PhotozParams` object with `n_phz` $= 2$ internally, this object is not used for spectroscopic SN data).

`n_sne`	Total number of SNe *per field.* You can just lump all the SNe in one field. Note that the SN Fisher matrix is always for a single field.
`dndz`	Type of the SN distribution in true redshift space (integer): 4 for D2k in Zhan et al. (2008), 5 for S20k in Zhan et al. (2008), 6 for $dn/dz =$ constant, 7 for a constant volume density, and 8 for SNAP SN distribution in Kim et al. (2004) *without local SNe.*
`sig_m`	Observed dispersion of the SN magnitude.
`n_f`	Number of fields (default 1).
`sig_f`	Undocumented. Always run `bin/snefm` with `sig_f` $= 0$.
`sne_FM`	Name of the SN Fisher matrix (default none).
`cal_FM`	Calculate the SN Fisher matrix if `cal_FM` is *true* (default *false*). Otherwise, `sne_FM` must exist for calculating the constraints.
`pri_evo`	Prior on the SN evolution parameters e_1 and e_2 (see below, default 0.015).
`pri_dz`	Prior on the photo-z bias parameters (default $0.1 \times$ `sig_z0`). The prior on the rms parameters is $\sqrt{2}$ times this. You can individualize the prior on each photo-z parameter in `src/snefm.cpp` if needed. `pri_dz` is not used if `n_phz` $= 0$.
`cmb_FM`	File name of the CMB Fisher matrix generated by `bin/cmbfm`, or any priors on cosmological parameters in the form of a 13×13 matrix (default none).
`pri_lnh`	Fractional prior on the Hubble constant (default 0.11).

Remarks The SN apparent magnitude is assumed to be (Zhan et al. 2008)

$$\bar{m}(z_{\rm p}) = \int \left[5\log D_{\rm L}(z) + M + e_1 z + e_2 z^2\right] p(z|z_{\rm p}) dz,$$

where the subscript p stands for photo-z, $D_{\rm L}$ is the luminosity distance, M is the SN absolute magnitude, e_1 and e_2 account for possible SN evolution, and $p(z|z_{\rm p})$ is the probability density of a SN at z given its photo-z $z_{\rm p}$. For spectroscopic surveys, $p(z|z_{\rm p})$ is a Dirac delta function $\delta^{\rm D}(z - z_{\rm p})$. The SN Fisher matrix is for a single field and does not include the priors. The order of parameters is: cosmological parameters as in the CMB Fisher matrix, photo-z rms parameters, photo-z bias parameters, M, e_1, and e_2.

5.3 WL and BAO

Program `bin/wlbao`

Purpose To generate galaxy and shear power spectra.

Syntax `wlbao parFile n_BAO n_WL n_phz n_b sig_z0 [dndz_z0 N_gal]`

`parFile` Name of the WL/BAO parameter file. Here is an example:

```
20000                    % SURVEY_AREA (deg^2)
3000    2000             % SWB_L_MAX WL_L_MAX
0  0  3.5  0.5  5.0      % DNDZ_MD Z_MIN Z_MAX Z_EXT Z_NO_GAL
z       u       u        % BAO_BIN_FLG WL_BIN_FLG PHZ_BIN_FLG
0.18    0.042            % WL_RMS_SHEAR WL_DRMSS_DZ
0       0                % not used
example/cmb/transf/tf    % TFFN
example/cmb/lcdm.par     % COSFN
example/wlbao            % OUT_DIR
2       wbcl/            % OUT_CL_FLG CLFN
0                        % not used
```

DNDZ_MD sets $n(z)$ type: 0 for $n(z) \propto z^2 e^{-z/z_0}$, 1 for $n(z) \propto z^2 e^{-z^2/z_0^2}$, and 2 for the one in Song & Knox (2004). Z_MIN is the minimum photo-z of the data and will be adjusted to be greater than 0.001 (src/gals/swab.cpp). For BAO, the minimum photo-z BAO_Z_MIN = max(0.15, Z_MIN). *_BIN_FLG sets the arrangement of bin widths or parameter intervals: z for width/interval $\propto 1 + z_\mathrm{p}$ or u for uniform. The galaxy rms shear γ_rms = WL_RMS_SHEAR + WL_DRMSS_DZ $\times z$. TFFN is the prefix of the transfer function names. COSFN is the name of the cosmological parameter file, and OUT_DIR is the output directory. OUT_CL_FLG sets the output mode: 0 for no output, or 2 for $P_{ij}(\ell)$ and derivatives in binary files. In the latter case, the galaxy/shear power spectra will be written with the prefix CLFN in the directory OUT_DIR/. See example/wlbao/wb.par and src/gals/swab.h for more details.

n_BAO	Number of BAO bins.
n_WL	Number of WL bins.
n_phz	Number of photo-z parameters in each component, i.e., n_phz photo-z bias parameters and n_phz photo-z rms parameters.
n_b	Number of galaxy bias parameters.
sig_z0	σ_{z0} as in $\sigma_z = \sigma_{z0}(1+z)$.
dndz_z0	z_0 in $n(z)$ (default 0.5).
N_gal	Number of galaxies per square arcmin (default 50).

Remarks The photo-z error distribution is simply a Gaussian with rms $\sigma_z = \sigma_{z0}(1+z)$ and bias $\delta z = 0$. To allow more freedom in the form of the distribution, you can modify the code to set the parameters $\sigma_{z,i}$ and δz_i individually.

To reduce the file size, which could be a few GB with many bins, the power spectra $P_{ij}(\ell)$ and their derivatives are output from $\ell = 2$ to $\min(\ell_{\max,i}, \ell_{\max,j})$. No noise is added to $P_{ij}(\ell)$, while the shot noise is included in the derivatives with respect to photo-z parameters. If the option NINJCL is enabled, the power spectra and derivatives contain the factor $\bar{n}_i \bar{n}_j$.

The spectra are sorted according to their ℓ_max in the binary files. OUT_DIR/CLFNbins.txt contains the information about the bins:

```
15 15 15 50 0.05
  5 G 0.150 0.298    67 1.04567792516
  ...
  0 S 0.001 0.701  2000 8.78085774288
  ...
 13 G 2.414 2.919  3000 3.63018440282
```

The first line is n_BAO + n_WL, n_phz, n_b, N_gal, and sig_z0. The columns of the rest are the bin ID, type (S for WL, and G for BAO), photo-$z_{i,\mathrm{beg}}$, photo-$z_{i,\mathrm{end}}$, $\ell_{\max,i}$, and $\bar{n}_i$ (arcmin^{-2}). Note that WL bins start from z_p = Z_MIN while BAO bins from BAO_Z_MIN (src/gals/swab.h).

OUT_DIR/CLFNphz.txt and OUT_DIR/CLFNgalb.txt are for the photo-z parameters (columns: z_i, δz_i, & $\sigma_{z,i}$) and galaxy bias parameters (columns: z_i & b_i), respectively.

Program bin/clcon

Purpose To calculate constraints on cosmological and other parameters from pre-calculated galaxy and shear power spectra.

Syntax
```
clcon parFile [pri_dz pri_b bao_AP wl_AP pri_MS use_BAO use_WL use_tom
      area l_min wl_l_ma no_nois cmb_FM z_max N_gal wb_FM]
```

`parFile`	Name of the WL/BAO parameter file, same as that of `bin/wlbao`.
`pri_dz`	Fractional prior on photo-z bias parameters (default 0.05), $\sigma_{\rm P}(\delta z_i) = $ `pri_dz` $\times \sigma_{z0}(1+z_i)$. For convenience, $\sigma_{\rm P}(\sigma_{z,i}) = \sqrt{2}\sigma_{\rm P}(\delta z_i)$.
`pri_b`	Fractional prior on galaxy bias parameters (default 100, i.e., no prior), $\sigma_{\rm P}(b_i) = $ `pri_b` $\times b_i$.
`bao_AP`	Amplitude of the galaxy additive noise power, i.e., $(A_i^{\rm g})^2$ (default 10^{-12}). Realistic values should be less than the power spectra themselves at $\ell \sim 1000$. I take `bao_AP` $< 10^{-7}$.
`wl_AP`	Amplitude of the shear additive noise power, i.e., $(A_i^\gamma)^2$ (default 10^{-14}). I take `WK_AP` $< 10^{-8}$.
`pri_MS`	Prior on the multiplicative shear errors f_i. I take `pri_MS` < 0.01.
`use_BAO`	Use galaxy auto power spectra if `use_BAO` is *true* (default *true*).
`use_WL`	Use shear power spectra if `use_WL` is *true* (default *true*). Include galaxy-shear power spectra as well if both `use_BAO` and `use_WL` are *true*.
`use_tom`	Use galaxy cross power spectra if `use_tom` is *true* (default *true*).
`area`	Area of the survey in deg^2 (default `SURVEY_AREA` in the parameter file).
`l_min`	Minimum ℓ, `CTOM_L_MIN` in `src/gals/ctom.h`, for both BAO and WL (default 40). See also `SWB_L_MIN` in `src/gals/swab.h` and `src/gals/ctom.cpp`.
`wl_l_ma`	`wl_L_MAX` (default 2000). For convenience, I set `SWB_L_MAX` = `wl_L_MAX`+1000 in `src/clcon.cpp`.
`no_nois`	No shot noise or systematics if `no_nois` is *true* (default *false*).
`cmb_FM`	File name of the CMB Fisher matrix (default `Planck.fsh`).
`z_max`	`Z_MAX` (default set in `parFile`). Bins beyond `z_max` will be discarded. The impact on dark energy constraints can be found in Zhan & Knox (2006b).
`N_gal`	Number of galaxies per square arcmin if `N_gal` > 0; otherwise it is discarded (default -1).
`wb_FM`	File name of the output Fisher matrix if present.

Remarks Command-line arguments override those in the WL/BAO parameter file, `parFile`. However, the values of `WL_L_MAX`, `SWB_L_MAX`, and `Z_MAX` will be adjusted not to exceed those in `parFile`. You can scale the survey with `area` and `N_gal`, but if the survey depth changes, you will have to adjust both `N_gal` and the redshift distribution of the galaxies. The scaling of `N_gal` works even if the power spectra are written with the $\bar{n}_i\bar{n}_j$ factor.

Pre-calculated galaxy and shear power spectra must have the prefix `OUT_DIR/CLFN`, and `cmb_FM` should be a properly formatted 13×13 Fisher matrix such as generated by `bin/cmbfm`.

You can set the priors on the photo-z parameters individually without assuming the (1+z) scaling. Same for the galaxy bias parameters. If `no_nois` is *true*, the photo-z parameters should be fixed.

The order of the parameters in the WL/BAO Fisher matrix is: cosmological parameters (same as CMB), galaxy clustering bias parameters, photo-z rms parameters, photo-z bias parameters, multiplicative shear errors, additive shear errors, and additive galaxy counting errors. Only relevant parameters will appear, e.g., no galaxy bias parameters if galaxy power spectra are not present.

5.4 Three-Dimensional BAO

Program `bin/bao`

Purpose To generate galaxy and shear power spectra.

Syntax `bao parFile cmb_FM` [`sig_z0 pri_dz pri_b pri_e pri_f pri_sn bao_FM cal_FM a_fac ng_fac`]

`parFile`	Name of the three-dimensional BAO parameter file. Here is an example:

```
example/cmb/lcdm.par      % COSFN
example/cmb/transf/tf     % TFFN
NumBAOBins 3
%id z_beg z_end   area      ng  bias  k_max
0     0.2   0.4  20000  4.9e-4  1.18  0.17
1     0.4   0.6  20000  8.1e-4  1.30  0.20
2     0.6   0.8  20000  4.4e-4  1.42  0.24
```

`COSFN` is the name of the cosmological parameter file, which should be the same as that used by `bin/cmbfm`. `TFFN` is the prefix of the transfer function names. `ng` is the comoving number density of galaxies in units of $h^3\mathrm{Mpc}^{-3}$. `k_max` is set to reduce the impact of nonlinear evolution (Seo & Eisenstein 2003). Other entries are self-explanatory.

`cmb_FM`	File name of the CMB Fisher matrix or any cosmological priors in the form of a 13×13 matrix.
`sig_z0`	σ_{z0} as in $\sigma_z = \sigma_{z0}(1+z)$ (default 0.05). Use 0 for a spectroscopic survey.
`pri_dz`	Prior on photo-z bias parameters (default 0.005): $\sigma_\mathrm{P}(\delta z_i) = \texttt{pri\_dz} \times (1+z_i)$. Note that it is defined differently from that for `bin/clcon`.
`pri_b`	Fractional prior on the galaxy bias parameters (default 100).
`pri_e`	Prior on the nuisance parameter e in the linear redshift distortion (default 0.05, see below).
`pri_f`	Prior on the nuisance parameter f in the linear redshift distortion (default 0.1, see below).
`pri_sn`	Prior on the shot noise (default 10^{10}).
`bao_FM`	File name of the BAO Fisher matrix.
`cal_FM`	Calculate the BAO Fisher matrix if `cal_FM` is *true* (default *false*). Otherwise, `bao_FM` must exist for calculating the constraints.
`a_fac`	Multiplicative factor to the area (default 1).
`ng_fac`	Multiplicative factor to the galaxy number density (default 1).

Remarks The linear redshift distortion factor is $(1+\beta\mu^2)^2$ (Kaiser 1987), where $\beta = \partial \ln G/\partial \ln a$ and $\mu = k_\parallel/k$. Scoccimarro (2004) has shown that the Kaiser formula is inaccurate even on very large scales ($\sim 10\%$ at $k \sim 0.1\, h\,\mathrm{Mpc}^{-1}$ for the monopole) and suggested a form of $1 + 2e(k)\mu^2 + f(k)\mu^4$. I simplify it with $1 + 2e\beta\mu^2 + f\beta^2\mu^4$, where $e = f = 1$ are scale independent.

The prior on the galaxy bias parameters does not matter much unless it is very strong, e.g., 1%. The shot noise is the reciprocal of the galaxy number density, so a prior on it is actually a prior on the comoving volume. I leave the shot noise to float.

The order of the parameters in the Fisher matrix is: cosmological parameters (same as CMB), galaxy bias parameters, photo-z rms parameters, photo-z bias parameters, e_i, f_i, and shot noise parameters.

If you want a two-stage forecast, turn on the option `BAO3DDH` and `make -C src clean bao`. The first stage parameters are "cosmological parameters," distance parameters, Hubble parameters, growth rate parameters, galaxy bias parameters, photo-z rms parameters, redshift distortion parameters e_i & f_i, and shot noise parameters. The cosmological parameters are in quotes because they no longer affect the distance and growth rate, which themselves are now parameters. An example is that the dark energy EOS parameters are not included in the first stage (more to come in next version). This is why you must instruct `bin/cmbfm` to generate the transfer functions for this purpose (§ 5.1). Do not forget to point `bao` to the correct transfer functions. See `example/bao3d/baodh.par`.

5.5 `zcos`

Program `bin/zcos`

Purpose To calculate the comoving distance $D(z)$, growth rate $G(z)$, growth index $f(z) = \partial \ln G/\partial \ln a$, Hubble parameter $H(z)$, time $t(z)$, $z(D)$, $z(G)$, $z(H)$, and $z(t)$.

Syntax `zcos [-dg0fhtar] z [OmegaM OmegaX w0 wa H0]`

flags	`-d` for D (h^{-1}Mpc), `-g` for G [G(0) = 1 regardless the cosmological model], `-0` for G_0 (growth rate at $z = 0$ in the other convention), `-f` for f, `-h` for H (km s^{-1} Mpc^{-1}), `-t` for t, `-a` for all of the above in that order, and `-r` for z from D, G, H, or t.
`z`	Redshift if `-r` is not present. Otherwise, one of {`d g h t`} should also be specified, and `z` will be taken as D, G, H, or t as indicated. If multiple flags in {`d g h t`} are specified with `-r`, the flag that appears earlier in the sequence (the `d g h t` sequence, not the order at the command-line) overrides those appear latter.
`OmegaM`	Matter fraction Ω_M (default 0.3).
`OmegaX`	Dark energy fraction Ω_X (default 0.7).
`w0`	Dark energy EOS parameter w_0 (default -1).
`wa`	Dark energy EOS parameter w_a (default 0).
`H0`	Hubble constant (default 71 km s^{-1}Mpc^{-1}).

6 CSWAB in More Depth

I discuss several useful components of CSWAB in this section for those who wish to customerize these components for other applications.

6.1 Global Variables

The global variables are resulted mostly from quick fixes to the ever-rising demand for new functionalities. Though convenient, they can be a hassle when one tries to modify the code. You have encountered many of them above. See `src/cmb/cmb.h`, `src/gals/swab.h`, `src/gals/angps.h`, and `src/gals/phzbin.h` for complete declarations and some brief explanations of the globals.

6.2 `class AdaptiveInterp`

This class is declared in `src/auxi/adpintp.h`. It is used to adaptively interpolate a function. The following two member functions initiate and setup up an `AdaptiveInterp` object:

```
  void Init( double a, double b, int n, double (*F)( double, void * ),
             void *p = NULL );
  void SetupTable( int n, double eRel, double eAbs = 0., double dxMi = 0.,
                   bool useMPI = true );
```

The arguments `a` and `b` bracket the interval over which interpolation is needed. `n` is the minimum number of evenly spaced points that will capture the major characteristics of the function `F()`, which takes a double and a pointer, e.g., `p`, to its parameters. Note that *no communication is allowed in* `F()`. `eRel` is the maximum fractional error of the interpolated function value anywhere, and `eAbs` is the maximum absolute error of the interpolated function integrated between `a` and `b`. `dxMi` is the minimum interval between two interpolation points. If `useMPI` is *true*, each computing node sets up a piece of the table and then collects all the pieces into a single table. If you have many tables to construct, it is more efficient to assign different tables to different nodes and then communicate the results explicitly than to divide each table among nodes and then communicate the results implicitly with `useMPI` = *true*.

Here is an example:

```
double FuncWrap( double x, void *params )
{
  double *ptr = (double *) params;
  double tD1 = (x - *ptr) / ptr[1];
  return  exp(-0.5 * tD1 * tD1) / ptr[1] / SQRT_2PI;
}
...
  double par[] = {1., 1.};
  AdaptiveInterp ai;

  ai.Init(-1., 3., 100, FuncWrap, par);
  ai.SetupTable(100, 1e-5, 0., 1e-4, false);
  cout << "FuncWrap(0.5) = " << FuncWrap(0.5, par)
       << "  Interp(0.5) = " << ai.Interpolate(0.5)
       << "  area = "        << ai.Integrate(-1., 3.) << endl;
  ai.WriteTable("spltab.txt");
```

The copy constructor and assignment operator have been implemented, so that you can make deep copies of an `AdaptiveInterp` object, e.g.,`ai1 = ai`, without worrying about the dynamically allocated elements. You can broadcast it from `nodeX` to other nodes in the communicator `comm` using `ai.Broadcast(nodeX, comm)`. You can write the table in `ai` to an `ofstream` object `oF` with `oF << ai << endl;`.

6.3 class CosParEvol

This class is declared in `src/cos/cospar.h`. It is used to calculate cosmological parameters and other quantities, such as the distance D and growth rate G, as functions of redshift. To (re)set the cosmological model for a `CosParEvol` object, call the member function

```
void Set( double Om, double Ox, double Ob, double eos0, double eosa,
          char *wFile, double h, double s8, double aMi, int tabLen );
```

where the arguments are Ω_M, Ω_X, Ω_B, w_0, w_a, the input file name of the dark energy EOS, h, σ_8, the minimum expansion factor of the interpolation table for G, D, & H, and the minimum length of the interpolation table (with `AdeptiveInterp`). If `wFile != NULL`, w_0 and w_a are ignored, and the dark energy EOS is quadratically interpolated from the file `wFile`. This file must contain two columns, the expansion factor ($0 \le a \le 1$) and the EOS. Entries do not have to be ordered, and there is no need to specify the number of entries. Separately, if $0 <$ `aMi` ≤ 0.99 *and* `tabLen` ≥ 3, all subsequent results of G, D, & H will be spline-interpolated in the range `aMi` $\leq a \leq 1$ until another `Set(...)` reverts the behavior. The following code fragment gives an example:

```
CPE  cpe;
cpe.Set(0.3, 0.7, 0.04, -1., 0., NULL, 0.73, 0.85, 0., 0);
distance = cpe.AngularDist_z(0.2, 1.0);  // D_A(z1, z2)
growth   = cpe.Growth_z(0.8);
hub_par  = cpe.Hubble_z(1.4);
curv_par = cpe.OmegaK_z(0.3);
```

where `CPE` is defined by `typedef CosParEvol CPE`. Note that the distance is always in units of h^{-1}Mpc and that the growth rate is normalized to $G(a = 1) = 1$ regardless the cosmological model. If you want to follow the convention that gives $G(a) = a$ for an Einstein-de Sitter universe but $G(a = 1) \neq 1$ in general, use `cpe.GetG0() * cpe.Growth_z(z)`.

The copy constructor and assignment operator have been implemented, so that you can make deep copies of a `CosParEvol` object, e.g., `cpe1 = cpe`, without worrying about the dynamically allocated elements. You can broadcast it from `nodeX` to other computing nodes in the communicator `comm` using `cpe.Broadcast(nodeX, comm)`. You can output the object with `cout << cpe << endl;`.

6.4 class CMBFisherMatrix

There is a useful global function worth mentioning in `src/cmb/cmb.h`:

```
void DCOS2CMBF( double cmbf[], COS_PAR i, DPAR_DIR dqDir );
```

As the name suggests, it translates the set of cosmological parameters to CMBFAST input parameters in the array `cmbf`. If `dqDir == DPAR_FID` or `i >= NUM_COS_PAR`, then `cmbf` contains the fiducial CMBFAST parameters. Otherwise, the `i`-th cosmological parameter will be perturbed in the direction `dqDir` (`DPAR_MINUS` or `DPAR_PLUS`) by the amount `DCOS[i]` before the whole set is converted to the CMBFAST parameters. The order of the CMBFAST parameters is w_0, w_a, Ω_C, Ω_B, Ω_X, Ω_ν, H_0, Y_p, τ, n_s, α_s, tensor-scalar ratio, and Δ^2_R. Note that nothing is done to Ω_ν and the tensor-scalar ratio.

The following two member functions of `CMBFisherMatrix` perform the essential tasks of initializing class members and calculating the Fisher matrix:

```
void Init( char prefix[], double f_sky );
void FisherMatrix( bool cal_Cls = false, char cmbfast[] = CMBF_EXEC );
```

where `prefix` is the prefix, e.g., the directory name, of the input/output CMB power spectra and transfer functions. The other arguments have the same meaning as those in the command-line arguments of `bin/cmbfm`. The constraints can be checked with `void Constraints(double cons[])`, and the Fisher matrix can be written to a file with `void WriteFM(char fName[])`.

6.5 class BAOWL

This class is declared in `src/gals/fisher.h`. It used to output the power spectra and derivatives. To start the calculations correctly, you need to call a series of member functions in the following order to initialize the members properly:

```
void InitCosmoParams();
void InitDnDz( double N_gal, double dndz_z0 );
void InitPowerSpectrum( char prefix[] );
void InitGalaxyBias( int i, double b0 = 1., double b1 = 0.84);
void InitPhotozParams( int i, double sz, double dz );
```

```
    void InitPhotozBins( int n_BAO, int n_WL );
    void InitBAOWL( double f_sky );
```

where `prefix` is the prefix of transfer function names, i.e., `TFFN` in `parFile`, and `f_sky` is the effective sky fraction covered by the survey. The arguments of `InitDnDz` and `InitPhotozBins` have the same meaning as those command-line arguments of `bin/wlbao`. The arguments of `InitGalaxyBias` are the number of parameters, $b(0)$, and $\mathrm{d}b/\mathrm{d}z$. The arguments of `InitPhotozParams` are the number of parameters in each component, σ_{z0}, and δz_0, where the photo-z rms and bias are assumed to be $\sigma_z = \texttt{sz} \times (1+z)$ and $\delta z = \texttt{dz} \times (1+z)$, respectively.

The galaxy/shear power spectra are calculated and written with

```
    void WriteAllCl( char dir[], char prefix[] = "" );
```

where `dir` is the output directory (`OUT_DIR` in `parFile`), and `prefix` is the prefix of the file names (`CLFN` in `parFile`).

6.6 class CosmicTomography

This class is declared in `src/gals/ctom.h`. It reads the galaxy/shear power spectra and calculates the constraints. The class members are initialized with

```
    void Init( char parFile[], int l_max, int wl_l_ma, int undocumented,
               bool use_BAO = true, bool use_WL = true,
               double z_max = 1e10, double N_gal = -1. );
```

where `l_max` is assigned to `SWB_L_MAX` and other arguments have the same meaning as the command-line arguments of `bin/clcon`. You can specify independent priors, such as the CMB Fisher matrix, with

```
    void SetCMB( char cmb_FM[], int order[], int dim );
```

where the Fisher matrix in the file `cmb_FM` has `dim` columns and `dim` rows, and `order` has `dim` elements. The `i`-th parameter in `cmb_FM` will be associated with the `order[i]`-th parameter in CSWAB (in case that `cmb_FM` is from another source and does not have the same order of the parameters), if $0 \le$ `order[i]` $<$ `NUM_COS_PAR`; it will be discarded if `order[i]` < 0; and it will be fixed if `order[i] >= NUM_COS_PAR && order[i] % dim == i`. Finally, the covariance matrix `cov` and Fisher matrix `wb_FM` are given by

```
    void CovFish( double cov[], double wb_FM[], double pri_dz, double no_use,
                  double pri_b, double pri_MS = 1e10, pri_WAi = 1e10,
                  double pri_BAi = 1e10, bool use_tom = true,
                  double wb_FM2[] = NULL ) const;
```

The dimensions of `cov` and `wb_FM` are given by `int NumParams()`. The array `cov` includes all the priors you have supplied, whereas `wb_FM` is only for WL/BAO. If `wb_FM2` $\neq$ `NULL`, `wb_FM2` will contain the WL/BAO Fisher matrix with the priors except `cmb_FM`. `pri_WAi` is the prior on the WL additive error parameters, which I take to be $\texttt{wl\_AP}^{1/2}$, and `pri_BAi` is for BAO. The rest arguments have the same meaning as the corresponding command-line arguments.

References

Cooray, A., & Sheth, R. 2002, Phys. Rep., 372, 1

Eisenstein, D. J., & Hu, W. 1999, ApJ, 511, 5

Hu, W., & Jain, B. 2004, Phys. Rev. D, 70, 043009

Huterer, D., Takada, M., Bernstein, G., & Jain, B. 2006, MNRAS, 366, 101

Jain, B., & Bertschinger, E. 1994, ApJ, 431, 495

Jeong, D., & Komatsu, E. 2006, ApJ, 651, 619

Kaiser, N. 1987, MNRAS, 227, 1

Kim, A. G., Linder, E. V., Miquel, R., & Mostek, N. 2004, MNRAS, 347,909

Knox, L., Song, Y.-S., & Zhan, H. 2006, ApJ, 652, 857 (astro-ph/0605536)

Ma, Z., Hu, W., & Huterer, D. 2006, ApJ, 636, 21

Peacock, J. A., & Dodds, S. J. 1996, MNRAS, 280, L19

Schneider, M., Knox, L., Zhan, H., & Connolly, A. 2006, ApJ, 651, 14 (astro-ph/0606098)

Scoccimarro, R. 2004, Phys. Rev. D, 70, 083007

Seo, H., & Eisenstein, D. J. 2003, ApJ, 598, 720

Seo, H.-J., & Eisenstein, D. J. 2005, ApJ, 633, 575

Song, Y.-S., & Knox, L. 2004, Phys. Rev. D, 70, 063510

Spergel, D. N. et al. 2007, ApJS, 170, 377

Springel, V. et al. 2005, Nature, 435, 629

White, M. 2005, Astropart. Phys., 24, 334

Zaldarriaga, M., & Seljak, U. 2000, ApJS, 129, 431

Zhan, H. 2006, J. Cosmol. Astropart. Phys., JCAP08(2006)008 (astro-ph/0605696)

Zhan, H., & Knox, L. 2006a, ApJ, 644, 663 (astro-ph/0509260)

Zhan, H., & Knox, L. 2006b, ArXiv e-prints, astro-ph/0611159

Zhan, H., Wang, L., Pinto, P., & Tyson, J. A. 2008, ApJ in press (arXiv:0801.3659)