Webnucleo Technical Report: Numerical Calculations in the wn_simple_gce Module

Bradley S. Meyer

April 12, 2013

The following presents some of the details behind the calculations in the wn_simple_gce Module. The module itself can be found at:

http://www.webnucleo.org/home/modules/wn_simple_gce/

1 Gas and Star Mass

wn_simple_gce follows the formalism of Clayton [1], who assumes a linear star formation rate, that is, a rate proportional to the mass of gas in the disk or solar annulus. Under the instantaneous recycling approximation, the rate of change of M_G, the mass of gas in the disk or the solar annulus, is

$$\frac{dM_G}{dt} = -\omega M_G + f(t), \tag{1}$$

where ω is the constant gas consumption rate and $f(t)$ is the rate of infall of metal-poor gas to build up the disk or annulus. The solution is [1]

$$M_G(t) = M_{G0} \exp\left[-\omega t + \int_0^t \frac{f(t')}{M_G(t')} dt'\right], \tag{2}$$

where M_{G0} is the initial gas mass. Clayton [2] defines the infall cycle number $\theta(t)$ to be

$$\theta(t) = \int_0^t \frac{f(t')}{M_G(t')} dt'. \tag{3}$$

The infall cycle number integrand is then $f(t)/M_G(t)$. The default infall cycle number integrand is that for Clayton's standard model 1 [2], which is given by

$$\frac{f(t)}{M_G(t)} = \frac{k}{t+\Delta}, \tag{4}$$

where k is an integer and Δ is a constant. For this default integrand, the infall cycle number function is

$$\theta(t) = k \ln\left(\frac{t+\Delta}{\Delta}\right). \tag{5}$$

The user may choose a different infall parameterization than the default. As of version 0.3, this is done with the wn_simple_gce API routine WnSimpleGce__updateInfallCycleFunctionsAndData(), which has the prototype:

```
void
WnSimpleGce__updateInfallCycleFunctionsAndData(
  WnSimpleGce *p_model,
  WnSimpleGce__infall_cycle_integrand pf_integrand,
  void *p_integrand_data,
  WnSimpleGce__infall_cycle_function pf_function,
  void *p_function_data
);
```

The inputs to this routine are:

- **p_model:** A pointer to a simple GCE model.
- **pf_integrand:** A pointer to a user-defined function computing the infall integrand $f(t)/M_G(t)$ at time t in Gyr.
- **p_integrand_data:** A pointer to a user-defined data structure containing extra data for the infall integrand.
- **pf_function:** A pointer to a user-defined function computing the infall function $\theta(t)$ at time t in Gyr.
- **p_function_data:** A pointer to a user-defined data structure containing extra data for the infall function.

The prototypes for the infall integrand and function are

```
double pf_integrand( double d_t, void *p_integrand_data );
```

and

```
double pf_function( double d_t, void *p_function_data );
```

where d_t is the time in Gyr. The user is required to supply the infall integrand but not the function. If the function is not supplied, the infall function $\theta(t)$ is computed numerically.

wn_simple_gce computes the gas mass from Eq. (2). If the infall cycle number function $\theta(t)$ is supplied, the gas mass is computed from it. If it is not, then $\theta(t)$ is computed numerically. The numerical calculation is done using the GNU Scientific Library (gsl) routine gsl_integration_qags(). Numerical integration proceeds until the solution reaches the relative tolerance D_TOLERANCE, which is set in WnSimpleGce.h. The maximum number of integration steps I_MAX_STEPS and the maximum allocated workspace I_WORKSPACE for the integrations are also set in WnSimpleGce.h.

All wn_simple_gce models start with zero mass in stars. The star mass builds up as gas converts into stars at the rate $\omega M_G(t)$. The star mass is computed numerically from

$$S(t) = \omega \int_0^t M_G(t')dt' \tag{6}$$

using the gsl_integration_qags() routine with the same tolerance, maximum steps, and workspace as for the gas mass.

Once the gas mass and star mass are available, the total mass in the disk or annulus is given by $M_{tot}(t) = M_G(t) + S(t)$. It should be clear that calculations for which the infall cycle number function $\theta(t)$ is provided will run faster than those in which $\theta(t)$ must be computed numerically.

2 Mass Fraction of Species

The mass fraction Z_i of a species i in the gas in a wn_simple_gce model is computed from the differential equation [1]

$$\frac{dZ_i}{dt} = y_i\omega - (Z_i - Z_{if})\frac{f(t)}{M_G(t)} - \lambda_i Z_i, \tag{7}$$

where y_i is the yield, Z_{if} is the mass fraction of species i in the infalling matter, and λ_i is the decay rate of species i. The general solution to this equation is

$$Z_i(t) = Z_i(0)e^{-\lambda_i t - \theta(t)} + \int_0^t dt' \left[y_i\omega + Z_{if}\frac{f(t')}{M_G(t')} \right] e^{-\lambda_i(t-t') - (\theta(t) - \theta(t'))}. \tag{8}$$

The yield y_i of species i is given by

$$y_i = \alpha_i + \beta_i Z_p + \gamma_i Z_p^2 + ... \tag{9}$$

where Z_p is the mass fraction of all primary isotopes at the given instant in time. The user may choose a yield for a species that is as large a polynomial in Z_p as desired by supplying the necessary yield coefficients. The total primary yield α is given by

$$\alpha = \sum_i \alpha_i, \tag{10}$$

the sum of the primary yields of all species. Thus, to compute the mass fraction of a species in the gas as a function of time, the user supplies the species yield coefficients and decay rate, the primary yield α, the initial gas mass and the gas consumption rate ω, the default infall rate parameters k and ω or his or her own infall rate function. The API also allows the user to provide the initial mass fraction of the species in the gas and functions to compute the primary metallicity and the mass fraction of the species in the infalling matter. If the initial mass fraction of the species and the infall primary metallicity and infall mass fraction functions are not specified, these quantities are all assumed to be zero. wn_simple_gce then computes the mass fraction of the species from Eq.

(8) by numerical integration using gsl_integration_qags() with the tolerance, maximum steps, and workspace defined as for other integrations in the module.

When the species is radioactive, the integrand can be essentially zero for much of the time range $(0, t)$. To handle this efficiently, wn_simple_gce integrates Eq. (8) backwards when the species is radioactive. In particular, the integral is computed as the sum

$$\int_0^t dt' ... = \int_{t-f/\lambda}^{t} dt' ... + \int_{t-2f/\lambda}^{t-f/\lambda} dt' ... + \int_{t-3f/\lambda}^{t-2f/\lambda} dt' ... + ..., \tag{11}$$

where f is a factor set by D_FACTOR in WnSimpleGce.h. The sum is continued until the term

$$\int_{t-(n+1)f/\lambda}^{t-nf/\lambda} dt' ... \tag{12}$$

contributes less than D_CONVERGE to the existing sum. At this point, the integral is taken to have converged. Of course, the lower limit to the integral can never be less than zero and is set to zero if $t - nf/\lambda < 0$.

As of version 0.5, it is possible for the user to define a species yield function that works in place of Eq. (9). To do this, the user writes a function with the prototype

```
double
my_species_yield_function(
  double d_primary_metallicity,
  void * p_data
);
```

In this function, *d_primary_metallicity* is the current primary metallicity Z_p and *p_data* is a pointer to a user-defined data structure. This function must return the yield for a species at the current metallicity. Once the user's function is defined, it can be set for a particular species with the API routine *WnSimpleGce__Species__updateYieldFunction()*. The data are updated with the API routine *WnSimpleGce__Species__updateUserYieldFunctionData()*.

As an example, consider a user-defined data structure

```
typedef struct {
  double dAlpha1;
  double dAlpha2;
  double dZBreak;
  double dWidth;
} user_yield_data;
```

The parameters in this structure are α_1, α_2, Z_{break}, and $width$. We consider these applying to a yield function

$$y_i(Z_p) = \alpha_1 + \frac{\alpha_2}{2}\left\{1 + erf\left(\frac{Z_p - Z_{break}}{width}\right)\right\}. \tag{13}$$

This corresponds to a yield that is primary with coefficient α_1 for $Z_p << Z_{break}$ and primary with coefficient $\alpha_1 + \alpha_2$ for $Z_p >> Z_{break}$. In between, the yield transitions from the former to the latter as Z_p nears Z_{break}. How rapidly this transition occurs with Z_p is given by *width*. This parameterization models, for example, the ^{56}Fe yield, which is primary and due to Type II supernovae early in the Galaxy's history ($y_i = \alpha_1$) and is then primary and due to both Type II and Type Ia later ($y_i = \alpha_1 + \alpha_2$). The relevant user yield function is then

```
double
my_yield_function(
  double d_primary_metallicity,
  user_yield_data * p_my_data
)
{

  return
    p_my_data->dAlpha1 +
    p_my_data->dAlpha2 *
    (
      1.
      +
      gsl_sf_erf(
        ( d_primary_metallicity - p_my_data->dZBreak )
        /
        p_my_data->dWidth
      )
    ) / 2.;

}
```

The user sets this function for species *WnSimpleGce__Species * p_species* with

```
WnSimpleGce__Species__updateYieldFunction(
  p_species,
  (WnSimple__Species__yield_function) my_yield_function
);
```

and updates the data with

```
WnSimpleGce__Species__updateUserYieldFunctionData(
  p_species,
  p_user_data
);
```

where *p_user_data* is the pointer to the user's data structure containing the necessary data. The chemical evolution model will now use *my_yield_function* and the currently set data to compute the yield for *p_species*.

The user may restore the default yield function for a species by setting the yield function pointer to NULL:

```
WnSimpleGce__Species__updateYieldFunction(
  p_species,
  NULL
);
```

References

[1] D. D. Clayton, *Galactic chemical evolution and nucleocosmochronology - Standard model with terminated infall*, Astrophys. J, 285 (1984), pp. 411–425.

[2] ——, *Galactic Chemical Evolution and Nucleocosmochronology: A Standard Model*, in Nucleosynthesis : Challenges and New Developments, 1985, pp. 65–88.