Supplemental Material

$^{40}Ar/^{39}Ar$ geochronology and the diffusion of ^{39}Ar in phengite-muscovite intergrowths during step-heating experiments *in vacuo*

Marnie Forster and Gordon Lister

Appendix C - C++ code for Arrhenius data

The following routines are C++ code from a class used to perform diffusivity calculations. The mathematics involved is self explanatory, derived from Crank (1975), Watson et al. (1933), cf McDougall and Harrison 1933.

```
double DiffusivityCalculation::FractionalLossCalculationForSlab ( double dt_a2 ) {
        double difference;
        double answer1, answer2;
        int n = 0;
        double z = 0.00, zz, z0, z1, z2, z3;
        zz = dt_a2 * 0.25;
        // if you like you can use an iterative summation
        while ( n <= iteration_count_for_summation ) {
                z0 = (2.0 * n + 1.0);
                z0 = z0 * z0;
                z1 = -z0 * pi_squared * zz;
                if (z1 < -1.0e9)
                        z2 = 0.00;
                else
                        z2 = exp (z1);
                z3 = (1.0 / z0);
                z = z + z2 * z3;
                n = n + 1.0;
        }
        answer1 = 1.0 - ( 8.0 / pi_squared ) * z;
        // or you can use an approximation
        double s1, s2;
        if ( zz < 1.0e-6)
                s1 = 1.00;
        else
                s1 = 1.00 - ( ( 8.0 / pi_squared ) * exp ( - pi_squared * zz ) );
        s2 = ( 2.00 / sqrt ( local_pi ) ) * sqrt ( dt_a2 );
        if ( s2 < 0.5 )
                answer2 =  s2;
        else
                answer2 =  s1;
return answer1;
}
```

```
double DiffusivityCalculation::FractionalLossCalculationForSphere ( double dt_a2 ) {
	double answer;
	double x;
	double z;
	double z1, z2, z3;
	z = 0.00;
	x = 1.0;
	z2 = 0.00;
	while ( x <= iteration_count_for_summation ) {
		z1 = - ( x * x) * pi_squared * dt_a2;
		if (z1 < -1.0e4)
			z2 = 0.00;
		else
			z2 = exp (z1);
		z3 = (1.0 / (x * x));
		z = z + z2 * z3;
		x = x + 1.0;
	}
	answer = 1.0 - (6.0 / pi_squared) * z;
	return answer;
}


#define watson_alpha_1	2.404825557695773
#define watson_fractional_release_transition_value	0.78
#define precision_in_watson_formula		0.001

double DiffusivityCalculation::FractionalReleaseForCylinder( double dt_a2 ) {
	double fractional_release;
	double alpha_squared = watson_alpha_1 * watson_alpha_1;
	double value1,  value2;
	double pi_value = atan( 1.00 ) * 4.0;
	// this is the answer if f < 0.78
	double z1 = ( 4.0 * sqrt ( dt_a2 / pi_value ) );
	double z2 = ( pow( dt_a2, 1.5 ) / ( 3.00 * sqrt ( pi_value ) ) );
	value1 =  z1 - dt_a2 - z2 - 0.2244122 * (dt_a2 * dt_a2 );
	// this is the answer if f > 0.78
	value2 = 1.00 - ( ( 4.00 / ( alpha_squared ) ) * exp( -alpha_squared * dt_a2 ) );
	if ( value2 > watson_fractional_release_transition_value )
		fractional_release = value2;
	else
		fractional_release = value1;
	return fractional_release;
}
```