KWW関数の計算方法について

June 23, 2000

by Y. Amo, Ph.D.
apj@atom.phys.ocha.ac.jp

Typeset using LaTeX(Mac pTeX2.1.8)

KWW 緩和の式を用いてスペクトルのフィッティングを行うために、与えられたパラメータと周波数領域で normalize された KWW 緩和の値を計算するプログラムを作成した。KWW 緩和は無限級数または積分であらわされ、パラメータと周波数範囲によって異なった式を用いる必要があった。

まず、級数を用いて計算した。条件によって収束しないことがあったので、積分を数値的に異なった方法で行った。この結果 romberg 積分を行うと良いことがわかった。級数と積分のそれぞれ良い部分をつなぎ合わせることで KWW 緩和の計算式を作り、Igor で動作する XFUNC にまとめた。

Contents

1 KWW 緩和の計算式

計算式は、"Further Cosiderations of Non Symmetrical Dielectric Relaxation Behabiour arising from a Simple Empirical Decay Function", G.Williams et.al, Trans. Faraday Soc., 67(1971), 1323-1335 に従った。

緩和時間の分布は、

$$\phi(t) = e^{-\left(\frac{t}{\tau_0}\right)^{\beta}} \tag{1}$$

で与えられる。ただし、$0 < \beta \leq 1$ である。

このとき、誘電率 $\varepsilon^* = \varepsilon(\omega) - i\varepsilon(\omega)$ は、

$$\frac{\varepsilon^*(\varepsilon) - \varepsilon_\infty}{\varepsilon_0 - \varepsilon_\infty} = \int_0^\infty \left[-\frac{d\phi(t)}{dt}\right] \exp(-i\omega t)\, dt \tag{2}$$

であらわされる。この積分は、$\lambda = \tau_0^\beta$、$s = \lambda_0 + i\omega$、$\lambda_0 \approx 0$ とおくことで

$$\frac{\varepsilon^*(\varepsilon) - \varepsilon_\infty}{\varepsilon_0 - \varepsilon_\infty} = \beta\lambda \int_0^\infty [\exp(-st)] \left[\exp\left(-\lambda t^\beta\right)\right] t^{\beta-1} dt \tag{3}$$

のようにかきなおすことができる。また、級数の形でもあらわすことができて、

$$\frac{\varepsilon^*(\varepsilon) - \varepsilon_\infty}{\varepsilon_0 - \varepsilon_\infty} = \sum_{n=1}^{\infty} (-1)^{n-1} \frac{1}{(\omega\tau)^{n\beta}} \frac{\Gamma(n\beta+1)}{\Gamma(n+1)} \left[\cos n\beta\frac{\pi}{2} - i\sin n\beta\frac{\pi}{2}\right] \tag{4}$$

と書ける。この級数が収束するのは、論文によれば、$0 < \beta \leq 0.25$ かつ $-4 \leq \log\omega\tau_0 \leq 4$ のときと、$0.25 < \beta < 1.0$ かつ $-1 \leq \log\omega\tau_0 \leq 4$ のときで、これ以外のときは、

$$\frac{\varepsilon^*(\varepsilon) - \varepsilon_\infty}{\varepsilon_0 - \varepsilon_\infty} = \sum_{n=1}^{\infty} (-1)^{n-1} \frac{(\omega\tau_0)^{n-1}}{\Gamma(n)} \Gamma\left(\frac{n+\beta-1}{\beta}\right) \left[\cos(n-1)\frac{\pi}{2} + i\sin(n-1)\frac{\pi}{2}\right] \tag{5}$$

を用いたとある。また（3）式は $t \to 0$ で $t^{\beta-1} \to \infty, (0 < \beta < 1)$ となる singularity を持つが、変数変換 $x = \lambda t^\beta$ により、積分範囲で解析的な形

$$\frac{\varepsilon^*(\varepsilon) - \varepsilon_\infty}{\varepsilon_0 - \varepsilon_\infty} = \int_0^\infty [\exp(-x)] \left[\exp\left(-i\omega\tau_0 x^{1/\beta}\right)\right] \tag{6}$$

になおすことができる。

2 級数の計算

まず、論文中の $\log(\omega\tau_0)$ が自然対数か常用対数か確認するため、(4) および (5) 式を用いて実部の計算を行った。

【結果】

この結果 (fig.2.1) と論文中の図を比較し、対数の底は 10 であることを確認した。 fig.2.1 では、一部収束していない部分があった。

(4),(5) 式の収束性を確認するために、$\log(\omega\tau_0)$ が $\pm$ 4 の範囲で、β の値を 0.1 から 1.0 まで 0.1 刻みに変化させて計算を行った (Fig.2.2)。計算で用いたプログラムは、IgorPro(WaveMetrics Inc.) の procedure として作成した。論文により、収束性のよいところでは n=10 で収束している。計算では、念のため n=50 とした。

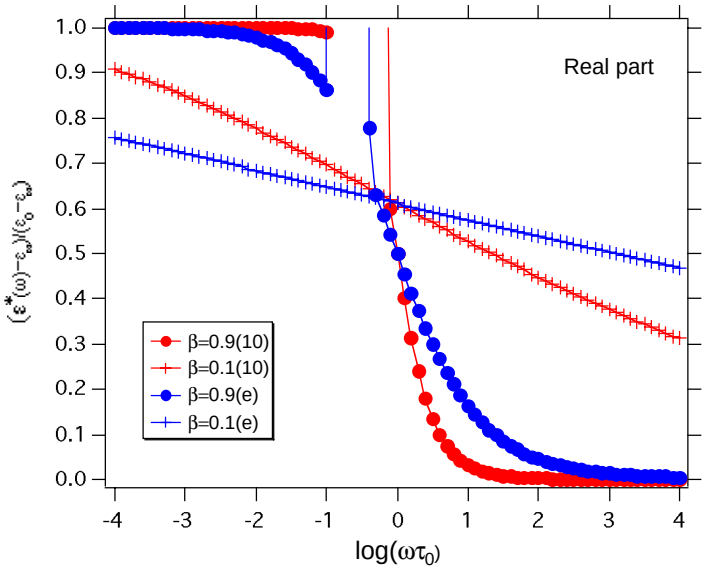

Figure 2.1: 対数の底の確認

級数中のガンマ関数は比の形で入っており、IgorPro にはガンマ関数の自然対数を求める関数が用意されている。このため、比の部分を対数の差にして、指数関数に代入することでガンマ関数の比の計算を行った。

【結果】

級数が収束する範囲は論文の記述通りではなかった。どちらの級数も発散する領域が存在し、その領域は β に依存した。

β=0.1 と β=0.2 では、計算範囲で (5) の級数が収束し、論文の図とほぼ同じ値になった。このことから、β が 0.2 以下では級数を使うだけでよいと考えられる。

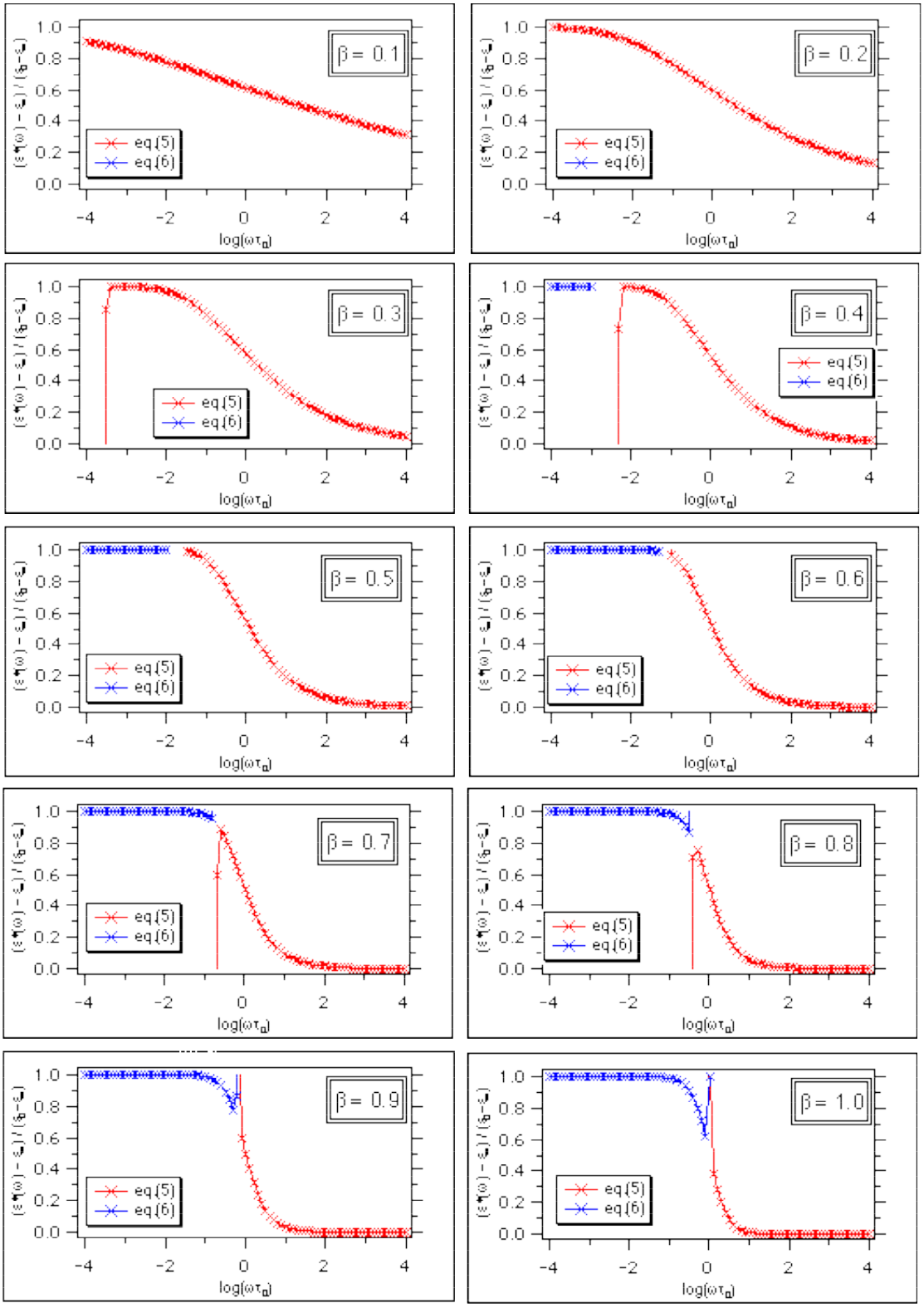

Figure 2.2: 収束性の確認

3 積分による評価

級数の計算で発散する領域をなくすために、(6) 式の積分を数値的に行った。計算方法は Simpson's Method, Romberg Integration (分割数 $(\frac{2}{3})\times 3^{n-1}$ および通常の分割) の 3 つを試した。それぞれ、Numerical Recipes のルーチンをそのまま用いた。計算の精度は、すべて `EPS = 1.0e-5` で行った。積分範囲は、x が 0 から 5 までと、0 から 5 0 までの 2 通りで行った。計算値を級数で求めた値に重ねて表示して収束性を確認した。計算は、 $\log(\omega\tau_0)$ が $\pm$ 4 の範囲で、β が 0.2 から 1.0 まで 0.1 きざみに行った。

プログラムの構成

Simpson's Method	simp.c,QSIMP.C,TRAPZD.C	
romberg1	romberg1.c, QROMO.C,MIDPNT.C,POLINT.C	その他
romberg2	romberg2.c,QROMB.C,POLINT.C,TRAPZD.C	その他

※大文字のファイル名は、Numerical Recipes のものをそのまま用いている。

【結果】

Simpson's Method は、収束する範囲が狭いうえに が大きくなると計算時間がかかるようになり、値を計算できる範囲が少なかった。グラフにはあまり出ていないが、これは正しい値に重なっているというよりは、計算できている範囲が少ないのであまり見えないことによる。

Romberg Ingtegration の収束性は良かったし、速やかに計算できる範囲も広かった。通常の刻み幅での積分 (romberg2) の方が計算結果の性質がよく、級数と同じ値に収束している。もっとも良かったのは、romberg2 で、x=50 まで計算したものである。実用的な速度で結果が出ているのは、

β	$\log(\omega\tau_0)$
0.2	-2 以下
0.3	-1 以下
0.4	0 以下
0.5 以上	1 以下

である。収束はもうすこし広い範囲であった。

積分の収束範囲は級数の (5) 式よりも広いため の小さい領域では romberg2 の数値積分を行い、大きい領域では級数 (4) 式を使うのがよさそうである。

romberg2 がもっとも良かったが、β=0.2 で実部がばらつくのと、β=0.4 で $\log(\omega\tau_0)$=1.0 の時の実部の値が変である。そこで QROMB.C 中の `EPS` の値を元の値 `1.0e-6` に戻して計算し、`1.0e-5` のときより良い結果を得た。

4 XFUNC の作成

計算方法がほぼわかったので、β と $\log(\omega\tau_0)$ の値に応じて、級数を使うか積分するかを決定し、KWW 関数の実部と虚部を計算する XFUNC を作った。 処理の振り分けは以下のようにした。

$0 < \beta \leq 0.2$		級数
$0.2 < \beta \leq 0.3$	$\log(\omega\tau_0) < -2$	積分
	$\log(\omega\tau_0) \geq -2$	級数
$0.3 < \beta \leq 0.4$	$\log(\omega\tau_0) < -1$	積分
	$\log(\omega\tau_0) \geq -1$	級数
$0.4 < \beta \leq 0.5$	$\log(\omega\tau_0) < 0$	積分
	$\log(\omega\tau_0) \geq 0$	級数
$0.5 < \beta \leq 1.0$	$\log(\omega\tau_0) < 1$	積分
	$\log(\omega\tau_0) \geq 1$	級数

級数は n=50 まで計算し、積分は Romberg Integration で `EPS=1.oe-6`、x が 0 から 50 まで数値積分している。

5 テスト

1. β が 0.1 から 1.0 まで 0.1 おきに計算した。
   【結果】発散無し。論文と見た目が同じ

2. β の計算のときに、ZoneRanger により IgorPro 内のメモリの状況を監視した。
   【結果】繰り返し関数を呼び出したときにメモリリークが起きないことを確認した。

3. 単一緩和のときの計算式の結果と β=1.0 の時の結果を比較した。
   【結果】重ねて表示し、表示している線の幅以内で一致することを確認した。

6 主なプログラム

使用した Numerical Recipes のファイルリスト

QSIMP.C
QROMB.C
QROMO.C
MIDPNT.C
POLINT.C
TRAPZD.C
GAMMLN.C

”NUMERICAL RECIPES in C 2nd Ed.”,Press et.al, Cambridge Univ. Press, ISBN 0-521-43108-5

XFUNC のソースリスト

KWWXFUNC.r	リソース
KWWXFUNC.h	
KWWXFUNC.c	XFUNC のエントリ部分
calcKWW.c	β と の値によって処理を振り分けて計算する

これに加えて、

QROMB.C	Romberg Integration
POLINT.C	補間
TRAPZD.C	積分の粗い値を出す
GAMMLN.C	ガンマ関数の対数を計算する

を使用している。