1 The Bingham closure

The Bingham closure is given by:

$$S(\mathbf{x}) = \int \psi_B(\mathbf{x},\mathbf{p})\mathbf{pppp}\,d\mathbf{p}, \tag{1}$$

where the Bingham distribution $\psi_B(\mathbf{x},\mathbf{p})$ is defined by the constraints that:

$$\int \psi_B(\mathbf{x},\mathbf{p})\,d\mathbf{p} = \phi(x), \tag{2a}$$

$$\int \psi_B(\mathbf{x},\mathbf{p})\mathbf{pp}\,d\mathbf{p} = D(\mathbf{x}), \tag{2b}$$

where ϕ and D are the zeroth and second moments with respect to the true distribution function ψ. We assume that $\psi_B(\mathbf{x},\mathbf{p})$ takes the form:

$$\psi_B(\mathbf{x},\mathbf{p}) = Ae^{B:\mathbf{pp}}. \tag{3}$$

Given $\phi(\mathbf{x})$ and $D(\mathbf{x})$, our goals is to find the coefficients A and B. Because B only appears contracted against a symmetric matrix, it is sufficient to assume that B is symmetric. In fact, we will see that our goal is even simpler than this: we will only be interested in computing $S : E$ and $S : D$, where $E = \nabla\mathbf{u} + \nabla\mathbf{u}^\intercal$. The purpose of this package is to provide optimized routines for computing these contractions. This documentation describes how the package works.

2 The Bingham Closure in 2D

2.1 A simple formula for the closure

From Chaubal and Leal, B and D are diagonalized in the same frame. We assume that we are in this frame; and compute the closure here. We will clean up details afterwards. In this frame, we have that:

$$1 = \int_0^{2\pi} Ae^{\lambda_0\cos^2\theta+\lambda_1\sin^2\theta}\,d\theta, \tag{4a}$$

$$\mu_0 = \int_0^{2\pi} Ae^{\lambda_0\cos^2\theta+\lambda_1\sin^2\theta}\cos^2\theta\,d\theta, \tag{4b}$$

$$\mu_1 = \int_0^{2\pi} Ae^{\lambda_0\cos^2\theta+\lambda_1\sin^2\theta}\sin^2\theta\,d\theta, \tag{4c}$$

Note we have assumed here that $\phi = 1$. μ and λ are the eigenvalues of D and B respectively, with $\mu_0 > \mu_1$. We note that λ can only be fixed up to an additive constant: to see this, let λ_0 and λ_1 solve the above equations. Then letting $\tilde{\lambda}_i = \lambda_i + C$ for $i = 0, 1$, we have:

$$\int_0^{2\pi} Ae^{\tilde{\lambda}_0\cos^2\theta+\tilde{\lambda}_0\sin^2\theta}f(\theta)\,d\theta = \int_0^{2\pi} Ae^{C}e^{\lambda_0\cos^2\theta+\lambda_1\sin^2\theta}f(\theta)\,d\theta, \tag{5}$$

which simply changes the definition of A. We can choose a convenient choice of C then; it is convenient to choose C so that $\lambda_0 + \lambda_1 = 0$. Then we have that:

$$1 = \int_0^{2\pi} Ae^{\lambda_0(\cos^2\theta-\sin^2\theta)}\,d\theta, \tag{6a}$$

$$\mu_0 = \int_0^{2\pi} Ae^{\lambda_0(\cos^2\theta-\sin^2\theta)}\cos^2\theta\,d\theta. \tag{6b}$$

Exploiting the trig identity that $\cos^2\theta - \sin^2\theta = \cos(2\theta)$, and because:

$$\int_0^{2\pi} e^{\lambda_0\cos(2\theta)}\,d\theta = 2\pi I_0(\lambda_0), \tag{7}$$

we find that

$$A = \frac{1}{2\pi I_0(\lambda_0)}, \tag{8}$$

where I_v is the modified Bessel function of the first kind. Thus we find: Now we're down to the single equation:

$$\mu_0 = \frac{1}{2\pi I_0(\lambda_0)} \int_0^{2\pi} e^{\lambda_0 \cos(2\theta)} \cos^2 \theta \, d\theta. \tag{9}$$

Evaluating this final integral and simplifying gives:

$$2\mu_0 = 1 + \zeta(\lambda_0), \tag{10}$$

where we have defined $\zeta(x) = I_1(x)/I_0(x)$. Thus to find λ_0 given μ_0, we simply need to solve this nonlinear equation for λ_0.

2.2 Solution of the closure equation and numerical issues

In this section we consider the issues with solving the nonlinear equation:

$$2\mu = 1 + \zeta(\lambda) \tag{11}$$

for λ, given $\mu \in [0.5, 1.0]$. Note that μ_0, the largest eigenvalue, must live in this range becuase it is the largest eigenvalue and the eigenvalues sum to 1. The fundamental problem with simply throwing a naive Newton solver at this equation is that as $\mu \to 1$, $\lambda \to \infty$. While not a problem in and of itself, $\zeta(\lambda)$ is the ratio of $I_1(\lambda)$ and $I_0(\lambda)$. Both of these functions diverge exponentially fast as λ gets large, and so naive evaluation of the ratio fails. Nevertheless, their ratio converges to 1: our primary challenge is to find a way to evaluate ζ stably for large arguments. Fortunately, we are in luck! As it turns out, we can write:

$$I_\nu(z) = \frac{e^z}{\sqrt{2\pi z}} \mathcal{P}_\nu(z), \tag{12}$$

where $\mathcal{P}_\nu(z)$ is a power series in z^{-1} that converges for sufficiently large z. Our strategy is clear, then. For small arguments, we can evaluate ζ directly. For larger arguments, we compute ζ by:

$$\zeta(\lambda) = \mathcal{P}_1(\lambda)/\mathcal{P}_0(\lambda). \tag{13}$$

In my numerical experiments, the power series $\mathcal{P}_0$ and $\mathcal{P}_1$ converge rapidly when the argument is at least 20, and direct evaluation has no issues for this size argument. We thus evaluate ζ directly for $|\lambda| \leq 20$, and indirectly via the power series representations for $|\lambda| > 20$.

In order to find λ, we compute the solution to the equation $1/2 + \zeta(\lambda)/2 - \mu = 0$. The Jacobian is given by:

$$2\mathcal{J}(\lambda) = 1 - \zeta(\lambda)/\lambda - \zeta(\lambda)^2. \tag{14}$$

Fortunately, the singularity in $\zeta(\lambda)/\lambda$ is removable. We construct a function to evaluate this quantity using the function_generator package, on an approximation interval of $[-1.001, 1.0]$. These bounds are chosen so that evaluation points for the Chebyshev interpolants used by the function_generator do not live at the singularity. When $|\lambda| > 1$, we evaluate this quantity direclty.

2.3 Fast evaluation of the closure equation

This provides a stable way to compute $\lambda(\mu)$ for every value of $\mu \in [0.5, 1.0]$, but it requires a Newton iteration for every value of μ given. Instead, we might consider constructing an interpolant for this function. Unfortunately as mentioned before, as $\mu \to 1$, $\lambda \to \infty$. Because the function_generator package allows for adaptive, brute force interpolation, it could probably handle this. But we can be smarter. Note that what we actually want to calculate is:

$$S(\mathbf{x}) = \psi_B(\mathbf{x}, \mathbf{p})\mathbf{pppp} \, d\mathbf{p}. \tag{15}$$

By exploiting identities, as we will show momentarily, we can reduce computing all of this to computing:

$$S_{0000}(\mathbf{x}) = \int_0^{2\pi} Ae^{\lambda(\mathbf{x})\cos(2\theta)} \cos^4\theta \, d\theta. \tag{16}$$

Again, this function can be computed analytically, and dropping the $\mathbf{x}$, the result is:

$$S_{0000} = \frac{1}{2} - \frac{\zeta(\lambda)}{4\lambda} + \frac{\zeta(\lambda)}{2} \tag{17}$$

Now we're getting somewhere: we just define the function $S_{0000}(\mu)$ as:

- Given, μ, solve Equation (11) for λ using a Newton iteration,
- Evaluate Equation (17) with the argument λ from the first step.

Fortunately, $S_{0000}(\mu)$ is a bounded and relatively smooth function of μ. Thus we can use function_generator to construct a nearly machine-precision and very accurate approximation of $S_{0000}(\mu)$.

3 The full algorithm

We now assume that we are given $D(\mathbf{x})$ and outline a method for computing $(S : E)(\mathbf{x})$ and $(S : D)(\mathbf{x})$. Because these computations are done pointwise, we will omit $\mathbf{x}$ for the remainder. We first compute the eigendecomposition of D:

$$D = \Omega\Lambda\Omega^\intercal. \tag{18}$$

Since D is symmetric positive-definite, we may apply the routine np.linalg.eigh (which is faster than np.linalg.eig) in order to find the ordered eigenvalues. We call μ_0 the larger eigenvalue, and μ_1 the smaller eigenvalue. Using the methodology from above, we compute $\tilde{S}_{0000}(\mu_0)$. Note that here it is denoted explicitly as $\tilde{S}_{0000}$ - this is because this is not actually S_{0000}, but that value in the diagonalized frame. Via identites, we may compute some of the other components as:

$$\begin{align}
\tilde{S}_{0011} &= \mu_0 - S_{0000}, \tag{19}\\
\tilde{S}_{1111} &= \mu_1 - S_{0011}, \tag{20}\\
\tilde{S}_{0001} &= 0, \tag{21}\\
\tilde{S}_{0111} &= 0, \tag{22}
\end{align}$$

We now simply have to perform a rotation:

$$S_{ijkl} = \Omega_{im}\Omega_{jn}\Omega_{kq}\Omega_{lp}\tilde{S}_{mnqp}, \tag{23}$$

to get S back. Calculuating this sum is somewhat nasty. Luckily we can exploit symmetries/identites to spped things up:

$$\begin{align}
S_{0000} &= \Omega_{00}^4\tilde{S}_{0000} + 4\Omega_{00}^3\Omega_{01}\tilde{S}_{0001} + 6\Omega_{00}^2\Omega_{01}^2\tilde{S}_{0011} + 4\Omega_{00}\Omega_{01}^3\tilde{S}_{0111} + \Omega_{01}^4 S_{1111}, \tag{24}\\
S_{0001} &= \Omega_{00}^3\Omega_{10}\tilde{S}_{0000} + (3\Omega^2\Omega_{01}\Omega_{10} + \Omega_{00}^3\Omega_{11})\tilde{S}_{0001} + 3(\Omega_{00}\Omega_{01}^2\Omega_{10} + \Omega_{00}^2\Omega_{01}\Omega_{11})\tilde{S}_{0011} + \tag{25}\\
&\quad (3\Omega_{00}\Omega_{01}^2 + \Omega_{11} + \Omega_{01}^3\Omega_{10})\tilde{S}_{0111} + \Omega_{01}^3\Omega_{11}\tilde{S}_{1111}. \tag{26}
\end{align}$$

Note that $\tilde{S}_{0001} = \tilde{S}_{0111} = 0$, and so these terms can be left out of the sums in implementation, reducing these rotations to:

$$\begin{align}
S_{0000} &= \Omega_{00}^4\tilde{S}_{0000} + 6\Omega_{00}^2\Omega_{01}^2\tilde{S}_{0011} + \Omega_{01}^4 S_{1111}, \tag{27}\\
S_{0001} &= \Omega_{00}^3\Omega_{10}\tilde{S}_{0000} + 3(\Omega_{00}\Omega_{01}^2\Omega_{10} + \Omega_{00}^2\Omega_{01}\Omega_{11})\tilde{S}_{0011} + \Omega_{01}^3\Omega_{11}\tilde{S}_{1111}. \tag{28}
\end{align}$$

Knowing these, we can exploit more identites to find the rest of the components:

$$S_{0000} + S_{0011} = D_{00}, \tag{29a}$$

$$S_{0100} + S_{0111} = D_{01}, \tag{29b}$$

$$S_{1100} + S_{1111} = D_{11}. \tag{29c}$$

Finally now that we have these components of S, we just need to compute $S : E$ and $S : D$. For a general symmetric tensor T, we have that:

$$(S : T)_{00} = S_{0000}T_{00} + S_{0011}T_{11} + 2S_{0001}T_{01}, \tag{30a}$$

$$(S : T)_{01} = S_{0001}T_{00} + S_{0111}T_{11} + 2S_{0011}T_{01}, \tag{30b}$$

$$(S : T)_{11} = S_{0011}T_{00} + S_{1111}T_{11} + 2S_{0111}T_{01}. \tag{30c}$$

Since $S : T$ is symmetric, $(S : T)_{10} = (S : T)_{01}$.