7 Numerical Methods

7.1 Optimization Methods

Some of the estimators discussed in the previous chapters cannot be derived analytically, but are implicitly defined as the solution to an optimization problem: $\hat{\theta} = \arg\min_{\theta \in \Theta} Q(\theta, Y^n)$. Chapter 5 dealt with the asymptotic properties of such estimators.[1] This section sketches the numerical algorithms that can be used to obtain $\hat{\theta}$. The goal is to give some intuition on the classes of methods available rather than to provide recipes to be implemented one-to-one, because every statistical software contains many such optimization methods as pre-programmed commands and advanced methods are available as external software.

Gradient-Based Iterative Algorithms Numerous optimization algorithms are based on an iterative updating of a current guess θ^m to obtain a new guess θ^{m+1} until, hopefully, the optimum is reached. They are motivated by the follwing mean-value expansion of the score of the objective function (vector of first-order derivatives), $Q^{(1)}(\theta, Y^n) = \frac{\partial Q(\theta, Y^n)}{\partial \theta}$, around θ^m:

$$Q^{(1)}(\theta^{m+1}, Y^n) \approx Q^{(1)}(\theta^m, Y^n) - Q^{(2)}(\theta^m, Y^n)(\theta^{m+1} - \theta^m) \ ,$$

where $Q^{(2)}(\theta, Y^n) = \frac{\partial^2 Q(\theta, Y^n)}{\partial \theta \partial \theta'}$ is the Hessian of the objective function (matrix of second-order derivatives). If $Q(\theta, Y^m)$ is a quadratic function, this relationship is exact. For θ^{m+1} to be the optimum – provided that it is interior – $Q^{(1)}(\theta^{m+1}, Y^n) = 0$ must hold. This yields the expression

$$\theta^{m+1} = \theta^m - Q^{(2)}(\theta^m, Y^n)^{-1} Q^{(1)}(\theta^m, Y^n) \ ,$$

which can be used to update guess θ^m to guess θ^{m+1}.

The Newton algorithm uses exactly this equation as-is:

[1] For this purpose, we wrote this (sample) objective function as Q_n to distinguish it from the population or limit objective function.

Algorithm 4 (Newton Algorithm)**.**

1. *Initialize θ^0 and specify a tolerance level $\epsilon > 0$.*
2. *For $m = 1, 2, ...,$ do the following:*
   (a) *Given θ^m, compute $Q^{(2)}(\theta^m, Y^n)^{-1}$ and $Q^{(1)}(\theta^m, Y^n)$ and set*

$$\theta^{m+1} = \theta^m - Q^{(2)}(\theta^m, Y^n)^{-1} Q^{(1)}(\theta^m, Y^n) \ .$$

   (b) *If $||(\theta^{m+1} - \theta^m|| < \epsilon$, take $\hat{\theta} = \theta^{m+1}$. Else, proceed to the next iteration.*

Instead of checking convergence based on the change in the parameter value θ from iteration m to iteration $m + 1$, it is also common to assesss the change in the objective function $Q(\theta, Y^n)$ or to check whether the score $Q^{(1)}(\theta, Y^n)$ is sufficiently close to zero.

If $Q(\theta, Y^m)$ was a quadratic function, the Newton algorithm would converge to the optimum in a single iteration. Even otherwise, the algorithm tends to work very well. A drawback of the Newton algorithm is that the inverse of the Hessian needs to be computed in every step, which is often computationally expensive. As a result, many other iterative optimization algorithms replace the Hessian with some other matrix that is easier to compute. For example, one might use the score squared or the expectation thereof, as motivated by the information matrix equality. The simplest approach is to just take the identity matrix, as in the Method of Steepest Descent.

Algorithms that build on the Newton algorithm also often specify a step size $r > 0$ that multiplies the updating-term $Q^{(2)}(\theta^m, Y^n)^{-1} Q^{(1)}(\theta^m, Y^n)$. The rationale is that taking $r = 1$ as in the Newton algorithm might i) imply too large updating steps, leading us to miss the optimum (obtain an increase in the objective function) or to cycle around the optimum, or ii) it might imply too small steps and therefore slow convergence. Some algorithms start every iteration with $r = 1$ and, if the objective function (or the absolute value of the score) increases, decrease r by enough such that it increases instead. Other algorithms choose r adaptively using some criterion, thereby updating r^m to r^{m+1} in every iteration.

Importantly, for non-convex objective functions, there is no guarantee that a numerical optimization procedure finds the global optimum. It is good practice to run the algorithm for many different initial values θ^0 and take the lowest value of all of these local minima as the final solution.

Iteration on Conditional Optimizers Sometimes, there is no closed form solution for the estimator entire parameter vector θ, but we can obtain analytical expressions for con-

ditional estimators (optimizers). For example, consider the GLS model from Section 3.4. Under Normality of errors, it leads to the likelihood

$$\begin{aligned} p(Y|\beta,\Sigma) &= \prod_{i=1}^{n}(2\pi\sigma_i^2)^{-\frac{1}{2}}exp\left\{-\frac{1}{2\sigma_i^2}(y_i - x_i'\beta)^2\right\} \\ &= (2\pi)^{-\frac{n}{2}}|\Sigma|^{-\frac{1}{2}}exp\left\{-\frac{1}{2}(Y - X\beta)'\Sigma^{-1}(Y - X\beta)\right\} , \end{aligned}$$

where $\Sigma = diag(\sigma_1^2, ..., \sigma_n^2)$.The joint estimator $\hat{\theta} = (\hat{\beta}, \hat{\Sigma}) = \arg\max_{\theta=(\beta,\Sigma)} p(Y|\beta,\Sigma)$ is not available analytically, but we can solve for the conditional estimators

$$\hat{\beta}|\Sigma = (X'\Sigma^{-1}X)^{-1}X'\Sigma^{-1}Y , \quad \text{and} \quad \hat{\Sigma}|\beta = diag\left\{(Y - X\beta)(Y - X\beta)'\right\} .^2$$

Note that the same conditional estimators are obtained under LS estimation:

$$\hat{\beta} = \arg\min_{\beta} \frac{1}{n}\sum_{i=1}^{n}(y_i - x_i'\beta)^2/\sigma_i^2 = \arg\min_{\beta} \frac{1}{n}(Y - X\beta)\Sigma^{-1}(Y - X\beta) ,$$

and in turn we set $\hat{\sigma}_i^2 = (y_i - x_i'\beta)^2$ for $i = 1 : n$, which corresponds to the above expression for $\hat{\Sigma}|\beta$.

One approach to obtain an estimator for $\hat{\theta} = (\hat{\beta}, \hat{\Sigma})$ is two-step GLS estimation, as discussed in Section 3.4:

1. Set $\Sigma = 1$ and let $\tilde{\beta}|(\Sigma = 1)$ be a preliminary estimate for β. Note that this amounts to OLS estimation.
2. Conditioning on this $\tilde{\beta}$ from the first step, compute $\tilde{\Sigma}|\tilde{\beta}$ and and use it to obtain the "feasible" GLS estimator $\hat{\beta}|\ \tilde{\Sigma}$. The final estimate of Σ is then given by $\hat{\Sigma}|\hat{\beta}$.

This estimator does not correspond to $\arg\max_{\beta,\Sigma} p(Y|\beta,\Sigma)$, but it still has good properties.[3]

We can obtain the estimator $\hat{\theta} = (\hat{\beta}, \hat{\Sigma}) = \arg\max_{\theta=(\beta,\Sigma)} p(Y|\beta,\Sigma)$ by iterating on the two conditional estimators above until convergence (see Meng and Rubin (1993)):

[2] i.e. $\hat{\sigma}_i^2 = (y_i - x_i'\beta)^2$.

[3] Recall from Chapter 3 that OLS is consistent even under heteroskedasticity, which in turn implies that $\tilde{\Sigma}$ is consistent (plug-in property). Finally, $\hat{\beta}|\ \tilde{\Sigma}$ is consistent thanks to consistency of $\tilde{\Sigma}$ (see Section 5.2.1), which in turn implies that $\hat{\Sigma}$ is consistent. The asymptotic variance of $\hat{\beta}$ needs to be adjusted for the fact that $\hat{\Sigma}$ is estimated (see Section 5.2.1).

Algorithm 5 (Meng and Rubin (1993) Algorithm for Two Parameter Groups, β and Σ)**.**

1. *Initialize* Σ^0 *– e.g.* $\Sigma^0 = I$ *– and specify tolerance level* $\epsilon > 0$*.*
2. *For* $m = 0, 1, 2, ...,$ *given* Σ^m*,*
   (a) *compute* $\beta^{m+1} = \hat{\beta}|\Sigma^m$*,*
   (b) *compute* $\Sigma^{m+1} = \hat{\Sigma}|\beta^{m+1}$*,*
   (c) *if* $||(\theta^{m+1} - \theta^m|| < \epsilon$*, take* $\hat{\theta} = \theta^{m+1}$*. Else, proceed to the next iteration.*

The Appendix states the algorithm for a generic parameter vector θ and a general partition of it.

7.2 Sampling Methods

Often in econometrics, we need to sample random numbers from a certain distribution. For example, to evaluate the properties of an estimator or test statistic in some model, we may want to sample data from the supposed distribution that characterizes the model (see Chapter 2 or Section 6.1). In addition, in Bayesian analysis, we oftentimes cannot obtain the posterior $p(\theta|Y)$ analytically, but rely on numerical sampling techniques. While sampling from standard distributions can be done easily and efficiently using commands from statistical software, we sometimes need to sample from some distribution specific to our application, e.g. a non-standard posterior. This section presents methods that we can use in these cases.

The random sampling commands in statistical software are based on a sample of uniform random numbers. It turns out that solutions to certain non-linear deterministic difference equations are indistinguishable from $U(0, 1)$ random numbers. Thereby, the so-called "seed" determines the initial condition of these difference equations and hence influences the (quasi-) random number we get. It is very important to set the seed at the beginning of any code that involves random sampling, as it ensures that the code always gives the same random numbers, which renders the analysis replicable.

Inverse CDF Method Suppose we want to draw a random number θ with pdf $f(\theta)$ and cdf $F(\theta)$. We can turn a Uniformly distributed random number, $\theta^* \sim U(0, 1)$, into such a random number with pdf $f(\theta)$ by setting $F(\theta) = \theta^*$ and solving for θ.