ISlide 1]

[**point to plot**] Suppose you want to obtain the area under this curve. An easy way to approximate this area is to break it into many tiny rectangles, trapezoids or other regions with known areas and sum them up.

Today, I'm going to demonstrate the power of Gaussian integration (technically, quadrature) to approximate areas by summing up as few as five such regions. In the process, I will give examples to Slide how this may be accomplished in Stata.

First, I'll introduce the concept of Gaussian quadrature in general, then I'll concentrate mostly on the simplest and most widely used case - Gauss Legendre quadrature. I'll follow this with some examples and then conclude with a short discussion of speed issues.

[Slide 2]. So what is Gaussian quadrature? Suppose we wish to numerically calculate the integral of some function H(x) over an interval (a, b). This can be done by first decomposing H(x) into a product of two other functions W and f and then approximating the integral by the weighted sum on the right, where the abcissas (x_j) and weights (w_j) depend on W(x) and usually a and b, but not on f.

With fixed and suitable choice of W and for a variety of f's, the approximation can be amazingly accurate with very few terms in the summation - far less than the number needed to achieve the same accuracy with summing rectangles or trapezoids.

[Slide 3]. So how do we choose W and obtain the abcissas and weights? The easiest way is to select W from among several widely-used standard forms. W should be chosen such that f is as smooth as possible..

[Slide 4]
i.e. so that f can be well approximated by a linear combination of M orthogonal polynomials $P_1(x)$ *to* $P_M(x)$. Recall that orthogonal polynomials satisfy this relationship [**point to it].**

However there are many sets of possible orthogonal polynomials – in Gaussian quadrature, the polynomials are specific to W and have particular names for standard W's.

For each of these standard W's in combination with standard integration intervals such as from zero to infinity, the abcissas and weights are well-known and may be obtained from published tables.

[Slide 5]. It turns out that the x_j's are actually the zeros of P_M....

[Slide 6]. ...and that the w's can be calculated from these quantities, which are easily computed since the P's are polynomials..[**point to it**].

[Slide 7]. Here are three of the more common type of Guassian quadrature – each with their own set of orthogonal polynomials. Which one should we use?

[Slide 8]. In Stata's -xtlogit-, the likelihood at each positive outcome is this intractable integral [**point to it]** where H(x) is this [**point to it]**. Mu is the logit of the probability of a positive outcome and in general would depend on various predictor variables not Sliden here. The objective is to estimate mu by maximum likelihood.

[Slide 9]. Note that the H(x) [**point to graph**] can be factored into e^{-cx^2} times a the smooth function f(x) [**point to graph**], and that the interval of integration is -∞ to ∞.

[Slide 10]. To summarize, we wish to **Evaluate** an integral over the whole real line, and the **Integrand** can be factored into e-^x2 times a smooth function

[Slide 11] ...therefore Guass-Hermite quadrature is called for –and that's what Stata does according to the V8 cross-sectional manual on p. 139. Also, see the section on –quadcheck- for some information on accuracy. This information is relevant not only for –xtlogit-, but also for other Stata analysis routines that use Gaussian quadrature.

[Slide 12]. This is an example of where Gauss-*Laguerre* quadrature would be effective. Note that H(x) [**point to graph**] has a vertical asymptote and would be difficult to integrate numerically. However by choosing W(x) to be $x^{-1/2}e^{-x}$, we can write H(x)=W(x)f(x) [**point to decomposition at bottom**] where f(x) [**point to graph**] is very well behaved. Thus the integral can be well-approximated by this sum [**point to sum at bottom right**].

[Slide 13]. From now on, I'll restrict discussion to the Gauss-Legendre case, where W(x) = 1.

[Slide 14]. This is the most flexible type of Gaussian quadrature since the same weights can be used for any interval of integration and the abcissas can be easily transformed from the standard ones over -1 to 1 to any finite interval.

[Slide 15]. Here's what the standard abcissas and weights look like for 5-pont and 10-point quadrature: - again, the abcissas Sliden here are for an integration range of –1 to + 1.

[Slide 16]. This illustrates how the standard abcissas get distributed proportionally for integration over an arbitrary interval – in this example, from –2.5 to + 3. The values of the weights are plotted against the y-axis as red dots.

[Slide 17] . Now I'll give a simple example to illustrate the accuracy of Gaussian integration.

[Slide 18]. Suppose we wish to evaluate the standard normal probability integral from 0 to 3 - norm(3) - norm(0) in Stata. The actual value to 10 decimal places, according to Abramowitz and Stegun's Handbook of Mathematical functions is this [**point to it**], and is the same as norm(3)-norm(0) to this to this many places.

[Slide 19]. Using Guassian quadrature with only 5 points as Sliden one obtains this value [**point to it**]

[Slide 20]. If you wanted to use Stata's –integ- command to approximate this integral over K equally spaced values of x, you would do something like this: [**point to statements in box]** But to achieve an accuracy similar to 5-point Gaussian quadrature, one would need K to be about 26 [**point to it**].

[Slide 21]. But how can we actually do Gaussian integration for each observation of a Stata data set? Here's a more complicated example:

[Slide 22]. Has anyone ever tried to fit a regression model to data such as this? If there is reason to believe the regression function is bounded between two limits (in this case zero and one), a plausible model is ...

[Slide 23]. ...this one where the regression function F(x) is an ogive shape – in particular, a Beta probability integral from 0 to x. Depending on the parameters p and q, a variety of response functions are possible, all increasing from zero to one as x increases.

[Slide 24]. Here's the data again. **[Slide 25]** .Suppose we wish to estimate p and q by nonlinear least squares with Stata's –nl- command.

[Slide 26]. To do nonlinear least squares, the user must write an –nl- program that evaluates E(y | x) – namely the integral Sliden, for each observation – each time the parameter estimates are updated. If *ibeta* didn't exist and we wanted to use Gaussian quadrature, how is this readily accomplished?

[**Slide 27]** One way to facilitate the integration is the so-called "wide' approach – merge in the abcissas and weights as extra variables – Sliden here in orange. It should be emphasized that all quadrature calculations should be done in double precision – so here, the weight and abcissa variables are all of the double precision type.

[**Slide 28]** After doing this, the data would look like this – the weight variables take on the same values for each observation, but the abcissa variables change because the range of integration depends on the value of x.

[Slide 29] Once these extra variables are merged with the original data, here's how one might write the –nl-program:

As many of you know, an –nl-program has an itiialization and an iteration section. This is what the initialization section might look like.

[**point to args**] – y is...

[**point to P=1, Q=1**] These are extremely naive first guesses at P and Q. I just did this to prevent clutter in this example program.

[**point to lnf..**] – For each value of *i*, lnf_i is going to be the log of the integrand f(p,q u_i) in the regression function and wf_i will be the product of the weight times f(p,q,u_i). These variables are initialized here so that they can be replaced later in the iteration section.

[**point to gaussrow**] – this do-file merges in the weights and abcissas from pre-existing datasets with weights and normalized abcissas for NP - the prescribed number of points. The normalized abcissas are converted to the correct ones for the 0 – X range of integration.

[Slide 30]. ..and here's the iteration section. For each quadrature point *i*, this program evaluates w_i*f(p,q,u_i), stores it in wf_i and sums across *i* using egen's row sum operator [**point to egen**].

Note that to evaluate f(p,q,u_i) [*Sliden at the bottom*] it is desirable to first evaluate its log. This is because it has [small and large components] . T

[**point to (b-a)*Eyx**] –After egen, the sum must be multiplied by _ the integration range. This is the complete approximation to the regression function for this iteration.

[Slide 31]. Here's the nonlinear regression results with 5-point Gaussian quadrature. The data was generated with P=2, Q=3 and sig=0.05.

[Slide 32]. Here's the same results, with the table at the bottom Slideing what happened when I tried 10-point, 20-point quadrature as well as direct evaluation using Stata's built in ibeta function. Note the results are almost identical.

[Slide 33]. Here's what the fits look like using 5-point quadrature and ibeta. The predicted regression functions are so close that you can't see them distinctly on the graph.

[Slide 34]. How can this example usage of Gaussian quadrature be speeded up and how does it compare with direct evaluation using *ibeta*?

[Slide 35] At the top is a fragment of code from the –nl- iteration section.

[Slide 36] One possible approach to speeding things up is to sum directly within the loop and not use –egen –

[Slide 37] Another thing to try is use only one lnf and wf variable and keep replacing it in the loop

[Slide 38] So here’s some timing results using the three methods: **e**gen, **s**umming separate variables and **s**umming one variable. The numbers Sliden are seconds to do the regression on an expanded data set of about 23000 observations.

[Slide 39].In conclusion, here are a few of the many available references on Gaussian quadrature.

The discussion in this one [point] in **Numerical Recipes** is an excellent introductory reference.

A more advanced discussion can be found in the **Wolfram Math World** site here [point to it].

Finally, tables of the abcissas and weights for various types of Gaussian quadrature are in the **Abramoviwits and Stegun Handbook of Mathematical Functions**. I used this to create my own Stata data sets for the 5, 10 and 20 point Gauss-Legendre cases. You can also use this to get equations for writing your own programs for generating the appropriate orthogonal polynomials, weights and abcissas for many types of Gaussian quadrature.