

DERIVEST

John R. D'Errico

Email: woodchips@rochester.rr.com

September 5, 2007

1 Introduction - Derivative Estimation

The general problem of differentiation of a function typically pops up in three ways in Matlab.

- The symbolic derivative of a function.
- Compute numerical derivatives of a function defined only by a sequence of data points.
- Compute numerical derivatives of an analytically supplied function.

Clearly the first member of this list is the domain of the symbolic toolbox, or some set of symbolic tools. Numerical differentiation of a function defined by data points can be achieved with the function gradient, or perhaps by differentiation of a curve fit to the data, perhaps to an interpolating spline or a least squares spline fit.

The third class of differentiation problems is where DERIVEST is valuable. This document will describe the methods used in DERIVEST.

2 Numerical differentiation of a general function of one variable

Surely you recall the traditional definition of a derivative, in terms of a limit.

$$f'(x) = \lim_{\delta \rightarrow 0} \frac{f(x + \delta) - f(x)}{\delta} \quad (1)$$

For small δ , the limit approaches $f'(x)$. This is a one-sided approximation for the derivative. For a fixed value of δ , this is also known as a finite difference approximation (a forward difference.) Other approximations for

the derivative are also available. We will see the origin of these approximations in the Taylor series expansion of a function $f(x)$ around some point x_0 .

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + \frac{(x - x_0)^2}{2}f''(x_0) + \frac{(x - x_0)^3}{6}f^{(3)}(x_0) + \frac{(x - x_0)^4}{24}f^{(4)}(x_0) + \frac{(x - x_0)^5}{120}f^{(5)}(x_0) + \frac{(x - x_0)^6}{720}f^{(6)}(x_0) + \dots \quad (2)$$

Truncate the series in (2) to the first three terms, then form the forward difference approximation (1), where $x = x_0 + \delta$.

$$f'(x_0) = \frac{f(x_0 + \delta) - f(x_0)}{\delta} - \frac{\delta}{2}f''(x_0) - \frac{\delta^2}{6}f'''(x_0) + \dots \quad (3)$$

When δ is small, δ^2 and any higher powers are vanishingly small. So we tend to ignore those higher powers, and describe the approximation in (3) as a "first" order approximation since the error in this approximation approaches zero at the same rate as the first power of δ .¹ The values of $f''(x_0)$ and $f'''(x_0)$, while unknown to us, are fixed constants as δ varies.

Higher order approximations arise in the same fashion. The central difference (4) is a second order approximation.

$$f'(x_0) = \frac{f(x_0 + \delta) - f(x_0 - \delta)}{2\delta} - \frac{\delta^2}{3}f'''(x_0) + \dots \quad (4)$$

3 Unequally spaced finite difference rules

While most finite difference rules used to differentiate a function will use equally spaced points, this fails to be appropriate when one does not know the final spacing. Adaptive quadrature rules can succeed by subdividing each sub-interval as necessary. But an adaptive differentiation scheme must work differently, since differentiation is a point estimate. DERI VEST generates a sequence of sample points that follow a log spacing away from the point in question, then it uses a single rule (generated on the fly) to estimate the desired derivative. Because the points are log spaced, the same rule applies at any scale, with only a scale factor applied.

¹We would normally write these additional terms using $O()$ notation, where all that matters is that the error term is $O(\delta)$ or perhaps $O(\delta^2)$, but explicit understanding of these error terms will be useful in the Romberg extrapolation step later on.

4 Odd and even transformations of a function

Returning to the Taylor series expansion of $f(x)$ around some point x_0 , an even function² around x_0 must have all the odd order derivatives vanish at x_0 . An odd function has all its even derivatives vanish from its expansion. Consider the derived functions $f_{odd}(x)$ and $f_{even}(x)$.

$$f_{odd}(x) = \frac{f(x - x_0) - f(-x - x_0)}{2} \quad (5)$$

The Taylor series expansion of $f_{odd}(x)$ has the useful property that we have killed off any even order terms, but the odd order terms are identical to $f(x)$, as expanded around x_0 .

$$\begin{aligned} f_{odd}(x) = (x - x_0)f'(x_0) + \frac{(x - x_0)^3}{6}f^{(3)}(x_0) + \\ \frac{(x - x_0)^5}{120}f^{(5)}(x_0) + \frac{(x - x_0)^7}{5040}f^{(7)}(x_0) + \dots \end{aligned} \quad (6)$$

Likewise, $f_{even}(x)$ has no odd order terms or a constant term, but other even order terms that are identical to $f(x)$.

$$f_{even}(x) = \frac{f(-x - x_0) - 2f(x_0) + f(x - x_0)}{2} \quad (7)$$

$$\begin{aligned} f_{even}(x) = \frac{(x - x_0)^2}{2}f^{(2)}(x_0) + \frac{(x - x_0)^4}{24}f^{(4)}(x_0) + \\ \frac{(x - x_0)^6}{720}f^{(6)}(x_0) + \frac{(x - x_0)^8}{40320}f^{(8)}(x_0) + \dots \end{aligned} \quad (8)$$

The point of these transformations is we can rather simply generate a higher order approximation for any odd order derivatives of $f(x)$ by working with $f_{odd}(x)$. Even order derivatives of $f(x)$ are similarly generated from $f_{even}(x)$. For example, a second order approximation for $f'(x_0)$ is trivially written in (9) as a function of δ .

$$f'(x_0; \delta) = \frac{f_{odd}(x_0 + \delta)}{\delta} - \frac{\delta^2}{6}f^{(3)}(x_0) \quad (9)$$

We can do better rather simply, so why not? (10) shows a fourth order approximation for $f'(x_0)$.

$$f'(x_0; \delta) = \frac{8f_{odd}(x_0 + \delta) - f_{odd}(x_0 + 2\delta)}{6\delta} + \frac{\delta^4}{30}f^{(5)}(x_0) \quad (10)$$

²An even function is one which expresses an even symmetry around a given point. An even symmetry has the property that $f(x) = f(-x)$. Likewise, an odd function expresses an odd symmetry, wherein $f(x) = -f(-x)$.

Again, the next non-zero term (11) in that expansion has a higher power of δ on it, so we would normally ignore it since the lowest order neglected term should dominate the behavior for small δ .

$$\frac{\delta^6}{252} f^{(7)}(x_0) \tag{11}$$

DERIVEST uses similar approximations for all derivatives of f up to the fourth order. Of course, its not always possible for evaluation of a function on both sides of a point, as central difference rules will require. In these cases, you can specify forward or backward difference rules as appropriate.

5 Romberg extrapolation methodology applied to derivative estimation

Some individuals might suggest that the above set of approximations are entirely adequate for any sane person. Can we do better?

Suppose we were to generate several different estimates of the approximation in (3) for different values of δ at a fixed x_0 . Thus, choose a single δ

5 ROMBERG EXTRAPOLATION METHODOLOGY APPLIED TO DERIVATIVE ESTIMATION

```
>> df1 = (f(del) - f(0))/del
df1 =
    1.05170918075648

>> df2 = (f(del/2) - f(0))/(del/2)
df2 =
    1.02542192752048

>> 2*df2 - df1
ans =
    0.999134674284488
```

In fact, this worked very nicely, reducing the error to roughly 1 percent of our initial estimates. Should we be surprised at this reduction? Not if we recall that last term in (3). We saw there that the next term in the expansion was $O(\delta^2)$. Since δ was 0.1 in our experiment, that 1 percent number makes perfect sense.

The Romberg extrapolant in (13) assumed a linear process, with a specific reduction in δ by a factor of 2. Assume the two term (linear + quadratic) residual term in (3), evaluating our approximation there with a third value of δ . Again, assume the step size is cut in half again. The three term Romberg extrapolant is given by (14).

$$f'_0 = \frac{1}{3}f'_\delta - 2f'_{\delta/2} + \frac{8}{3}f'_{\delta/4} \quad (14)$$

A quick test in matlab yields much better results yet.

```
>> format long g
>> f = @(x) exp(x);
>> del = 0.1;

>> df1 = (f(del) - f(0))/del
df1 =
    1.05170918075648

>> df2 = (f(del/2) - f(0))/(del/2)
df2 =
    1.02542192752048

>> df3 = (f(del/4) - f(0))/(del/4)
df3 =
    1.01260482097715

>> 1/3*df1 - 2*df2 + 8/3*df3
ans =
    1.00000539448361
```

Again, DERI VEST uses the appropriate multiple term Romberg extrapolants for all derivatives of f up to the fourth order. This, combined with

the use of high order approximations for the derivatives, allows the use of quite large step sizes.

6 Uncertainty estimates for DERIVEST

We can view the Romberg extrapolation step as a polynomial curve fit in the step size parameter δ . Our desired extrapolated value is seen as simply the constant term coefficient in that polynomial model. Remember though, this polynomial model (see (10) and (11)) has only a few terms in it with known non-zero coefficients. That is, we will expect a constant term a_0 , a term of the form $a_1\delta^4$, and a third term $a_2\delta^6$.

A neat trick to compute the "statistical" uncertainty in the estimate of our desired derivative is to use statistical methodology for that error estimate. While I do appreciate that there is nothing truly statistical or stochastic in this estimate, the approach still works nicely, providing a very reasonable estimate in practice. A three term Romberg-like extrapolant, then evaluated at four distinct values for δ , will yield an estimate of the standard error of the constant term, with one spare degree of freedom. The uncertainty is then derived by multiplying that standard error by the appropriate percentile from the Students-t distribution.

```
>> tcdf(12.7062047361747, 1)
ans =
           0.975
```

This critical level will yield a two-sided confidence interval of 95 percent.

These error estimates are also of value in a difference sense. Since they are efficiently generated at all the different scales, the particular spacing which yields the minimum predicted error is chosen as the best derivative estimate. This has been shown to work consistently well. A spacing too large tends to have large errors of approximation due to the finite difference schemes used. But a too small spacing is bad also, in that we see a significant amplification of least significant fit errors in the approximation. A middle value generally seems to yield quite good results. For example, DERIVEST will estimate the derivative of e^x automatically. As we see, the final overall spacing used was 0.1953125.

```
>> [d, e, del] = derivest(@(x) exp(x), 1)
d =
           2.71828182845904
e =
           1.02015503167879e-14
del =
           0.1953125
```

However, if we force the step size to be artificially large, then approximation error takes over.

```
>> [d, e, del] = derivest(@(x) exp(x), 1, 'FixedStep', 10)
d =
      2.3854987890005
e =
      3.90016042034995
del =
      10
```

And if the step size is forced to be too small, then we see noise dominate the problem.

```
>> [d, e, del] = derivest(@(x) exp(x), 1, 'FixedStep', .0000000001)
d =
      2.71826406220403
e =
      0.000327191484277048
del =
      1e-10
```

DERIVEST, like Goldilocks in the fairy tale bearing her name, stays comfortably in the middle ground.

7 DERIVEST in action

How does DERIVEST work in action? A simple nonlinear function with a well known derivative is e^x . At $x = 0$, the derivative should be 1.

```
>> [d, err] = derivest(@(x) exp(x), 0)
d =
      0.999999999999997
err =
      2.22066469352214e-14
```

A second simple example comes from trig functions. The first four derivatives of the sine function, evaluated at $x = 0$, should be respectively $[\cos(0), -\sin(0), -\cos(0), \sin(0)]$, or $[1, 0, -1, 0]$.

```
>> d = derivest(@(x) sin(x), 0, 1)
d =
      0.999999999999999
>> d = derivest(@(x) sin(x), 0, 2)
```

```
d =  
    0  
  
>> d = derivest(@(x) sin(x), 0, 3)  
d =  
    -1.0000000000000046  
  
>> d = derivest(@(x) sin(x), 0, 4)  
d =  
    0
```

8 Gradient (**GRADEST**) and Hessian (**HESSIAN**) estimation

Estimation of the gradient vector (**GRADEST**) of a function of multiple variables is a simple task, requiring merely repeated calls to **DERIVEST**. Likewise, the diagonal elements of the hessian matrix are merely pure second partial derivatives of a function. **HESSDIAG** accomplishes this task, again calling **DERIVEST** multiple times. Efficient computation of the off-diagonal (mixed partial derivative) elements of the Hessian matrix uses a scheme much like that of **DERIVEST**, wherein **DERIVEST** is called to determine an initial step size, then Romberg extrapolation is used to improve a set of second order finite difference estimates of those mixed partials.

9 Conclusion

DERIVEST is an adaptive scheme that can compute the derivative of arbitrary (well behaved) functions. It is reasonably fast as an adaptive method. Many options have been provided for the user who wishes the ultimate amount of control over the estimation.

10 Acknowledgments

My thanks are due to Shaun Simmons for convincing me to learn enough LaTeX to write this document.

References

- [1] Lyness, J. M., Moler, C. B. (1966). Vandermonde Systems and Numerical Differentiation. *Numerische Mathematik*.

REFERENCES

REFERENCES

- [2] Lyness, J. M., Moler, C. B. (1969). Generalized Romberg Methods for Integrals of Derivatives. *Numerische Mathematik*.
- [3] *NAG Library*. NAG Fortran Library Document: D04AAF