Math 3607: Homework 11

Selected Solutions

1. (Optimal Step Size)

   (b) Let V_h be an m-th order finite difference formula for $f'(x)$, that is,

   $$V_h = f'(x) + ch^m + O(h^{m+1}), \quad \text{for some constant } c, \tag{1}$$

   and let $\widehat{V}_h$ be the floating-point evaluation of V_h. Assuming that round-off errors only occur in evaluation of f, by a similar argument as shown in lecture, one can show that

   $$\widehat{V}_h = V_h + M\frac{\boxed{\text{eps}}}{h} + O(\boxed{\text{eps}}), \quad \text{for some constant } M. \tag{2}$$

   By (1) and (2), the total error in approximating $f'(x)$ via V_h on a computer is

   $$\widehat{V}_h - f'(x) = \underbrace{ch^m + O(h^{m+1})}_{\text{truncation error}} + \underbrace{M\frac{\boxed{\text{eps}}}{h} + O(\boxed{\text{eps}})}_{\text{round-off error}}.$$

   The leading error term

   $$g(h) := ch^m + M\frac{\boxed{\text{eps}}}{h}$$

   is minimized when $g'(h) = 0$, which occurs when $h^{m+1} \approx \boxed{\text{eps}}$. So the total error is minimized when h is set to be approximately

   $$h \approx \boxed{\text{eps}}^{\frac{1}{m+1}},$$

   in which case the leading error term is $O(\boxed{\text{eps}}^{\frac{m}{m+1}})$, because

   $$\begin{aligned} g(\boxed{\text{eps}}^{\frac{1}{m+1}}) &= c\left(\boxed{\text{eps}}^{\frac{1}{m+1}}\right)^m + M\boxed{\text{eps}}^{1-\frac{1}{m+1}} \\ &= (c+M)\boxed{\text{eps}}^{\frac{m}{m+1}} = O\left(\boxed{\text{eps}}^{\frac{m}{m+1}}\right). \end{aligned}$$

   (c) Below is a program which approximates a Jacobian by 1st-order forward difference. Note that $h = \sqrt{\boxed{\text{eps}}}$ is used inside the program.

```
function J = jacfd(f, x0)
% JACFD Approximation of a Jacobian by 1st-order forward difference
% Input:
%   f      function to be differentiated
%          which takes (n-by-1) column vector as an input
%          and produces (m-by-1) column vector as an output
%   x0     evaluation point
% Output:
```

```
%   J      approximate Jacobian (m-by-n)

    h = sqrt(eps);    % optimal step size
    y0 = f(x0);       % evaluate f(x0) once and for all
    m = length(y0);   % see specification above
    n = length(x0);   % see specification above
    J = zeros(m,n);   % preallocation (for efficient memory alloc)
    I = eye(n);
    for j = 1:n
        J(:,j) = ( f(x0+h*I(:,j)) - y0 ) / h; % FD formula
    end
end
```