UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING
Department of Electrical &
Computer Engineering

ECE 204 *Numerical methods*

Newton's method for finding extrema in *n* dimensions

Douglas Wilhelm Harder, LEL, M.Math.
dwharder@uwaterloo.ca
dwharder@gmail.com

CC BY NC SA

Introduction

- In this topic, we will
  - Describe using Newton's method to find extrema of real-valued functions of a vector variable
  - Review the concept of the gradient and the Jacobian of a vector-valued function of a vector variable
  - Look at two examples
  - Understand when to use this technique

Newton's method

- Here is a straight-forward idea:
  - Apply Newton's method to the gradient

- Issue: You're not sure if the point is a maximum, a minimum, or a saddle point
  - A saddle point is where all partial derivatives are zero, but the function does not achieve an extremum

Newton's method

- At an extreme point, all the partial derivatives are zero
  - Consequently, we must find a point such that

$$\vec{\nabla} f(\mathbf{u}) = \mathbf{0}$$

- You will recall that

$$\vec{\nabla} f(\mathbf{u}) \stackrel{\text{def}}{=} \begin{pmatrix} \frac{\partial}{\partial u_1} f(\mathbf{u}) \\ \vdots \\ \frac{\partial}{\partial u_n} f(\mathbf{u}) \end{pmatrix}$$

Newton's method

- Thus, $\vec{\nabla} f(\mathbf{u}) = \mathbf{0}$ becomes a system of *n* equations in *n* unknowns
  - If *f* is a non-constant linear polynomial in **u**, then there are no solutions
  - If *f* is a quadratic polynomial in **u**, then the gradient defines a system of linear equations: just use linear algebra
  - Otherwise, the gradient is non-linear, so go back to our algorithm on Newton's method in *n* dimensions: we can find a solution to $\mathbf{g}(\mathbf{u}) = \mathbf{0}$ by choosing an initial $\mathbf{u}_0$ and iterating by solving $\mathbf{J}(\mathbf{g})(\mathbf{u}_k)\Delta\mathbf{u}_k = -\mathbf{g}(\mathbf{u}_k)$ and setting $\mathbf{u}_{k+1} \leftarrow \mathbf{u}_k + \Delta\mathbf{u}_k$
- In this case, our vector-valued function of a vector variable is the gradient of *f*, so solve $\mathbf{J}(\vec{\nabla} f)(\mathbf{u}_k)\Delta\mathbf{u}_k = -\vec{\nabla} f(\mathbf{u}_k)$ and set

$$\mathbf{u}_{k+1} \leftarrow \mathbf{u}_k + \Delta\mathbf{u}_k$$

The Hessian

- The Jacobian of the gradient of a real-valued function of a vector variable is called the *Hessian* of the function:

$$\mathbf{J}\left(\vec{\nabla} f\right)(\mathbf{u}) \overset{\text{def}}{=} \mathbf{H}(f)(\mathbf{u})$$

  - We will continue to use the notation with the Jacobian

- If the function is sufficiently differentiable,
  the Hessian is symmetric

The Hessian

- This is the Hessian of f :

$$\mathbf{J}\left(\vec{\nabla} f\right)(\mathbf{u}) \overset{\text{def}}{=} \mathbf{H}(f)(\mathbf{u}) \overset{\text{def}}{=} \begin{pmatrix} \frac{\partial^2}{\partial u_1^2} f(\mathbf{u}) & \frac{\partial^2}{\partial u_2 \partial u_1} f(\mathbf{u}) & \cdots & \frac{\partial^2}{\partial u_n \partial u_1} f(\mathbf{u}) \\ \frac{\partial^2}{\partial u_1 \partial u_2} f(\mathbf{u}) & \frac{\partial^2}{\partial u_2^2} f(\mathbf{u}) & \cdots & \frac{\partial^2}{\partial u_n \partial u_2} f(\mathbf{u}) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2}{\partial u_1 \partial u_n} f(\mathbf{u}) & \frac{\partial^2}{\partial u_2 \partial u_n} f(\mathbf{u}) & \cdots & \frac{\partial^2}{\partial u_n^2} f(\mathbf{u}) \end{pmatrix}$$

$$\vec{\nabla} f(\mathbf{u}) \overset{\text{def}}{=} \begin{pmatrix} \frac{\partial}{\partial u_1} f(\mathbf{u}) \\ \vdots \\ \frac{\partial}{\partial u_n} f(\mathbf{u}) \end{pmatrix}$$

The Hessian

- Recall from applying Newton's method to the derivative of a real-valued function of a real variable:
  - A solution to $f^{(1)}(x) = 0$ is
    - A local minimum if $f^{(2)}(x) > 0$
    - A local maximum if $f^{(2)}(x) < 0$
    - Of unknown character if $f^{(2)}(x) = 0$
      - That is, it could be a maximum, a minimum or a saddle point
- The same applies here:
  - A solution to $\vec{\nabla} f\left(\mathbf{u}\right) = \mathbf{0}$ is
    - A local minimum if $\mathbf{J}\left(\vec{\nabla} f\right)\left(\mathbf{u}\right)$ is *positive definite*
      - That is, all eigenvalues are positive
    - A local maximum if $\mathbf{J}\left(\vec{\nabla} f\right)\left(\mathbf{u}\right)$ is *negative definite*
      - That is, all eigenvalues are negative
    - Of unknown character otherwise

Example

- Consider the function

$$5x^2 + 6y^2 + 7z^2 + 2xy - 3yz - 28x + 6y - 3z + 1$$

  - This is a quadratic polynomial in **u**, so we calculate:

$$\mathbf{u} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$\vec{\nabla} f(\mathbf{u}) = \begin{pmatrix} 10x + 2y - 28 \\ 12y + 2x - 3z + 6 \\ 14z - 3y - 3 \end{pmatrix} \qquad \mathbf{J}\left(\vec{\nabla} f\right)(\mathbf{u}) = \begin{pmatrix} 10 & 2 & 0 \\ 2 & 12 & -3 \\ 0 & -3 & 14 \end{pmatrix}$$

  - Thus, all we must solve is:

$$\mathbf{J}\left(\vec{\nabla} f\right)(\mathbf{0})\mathbf{u} = -\vec{\nabla} f(\mathbf{0}) \quad \text{or} \quad \begin{pmatrix} 10 & 2 & 0 \\ 2 & 12 & -3 \\ 0 & -3 & 14 \end{pmatrix} \mathbf{u} = \begin{pmatrix} 28 \\ -6 \\ 3 \end{pmatrix}$$

  - Thus, $\mathbf{u} = \begin{pmatrix} 3 \\ -1 \\ 0 \end{pmatrix}$, we note $\vec{\nabla} f(\mathbf{u}) = \mathbf{0}$, and $f(\mathbf{u}) = -44$

Example

- Consider the function

$$5x^2 + 6y^2 + 7z^2 + 2xy - 3yz - 28x + 6y - 3z + 1$$

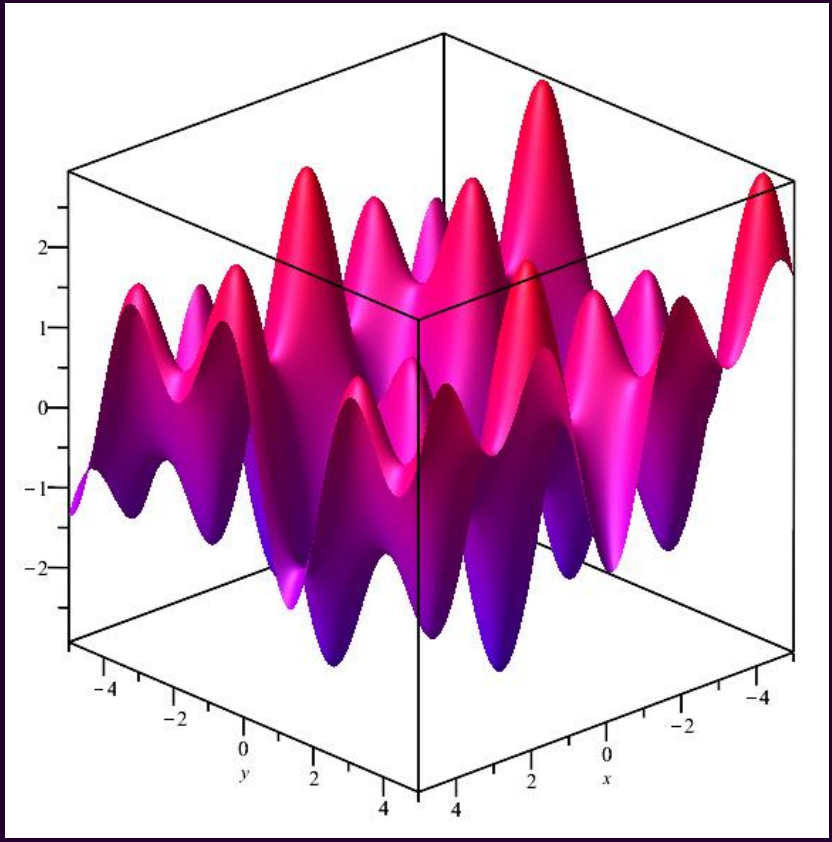
  - Recall if you applied Newton's method to a system of linear equations, it would converge after one iteration

Example

- Consider the function

$$\sin(2x + 5) + \sin(y - 3) + \sin(x - 2y - 4)$$

$$\mathbf{u} = \begin{pmatrix} x \\ y \end{pmatrix}$$

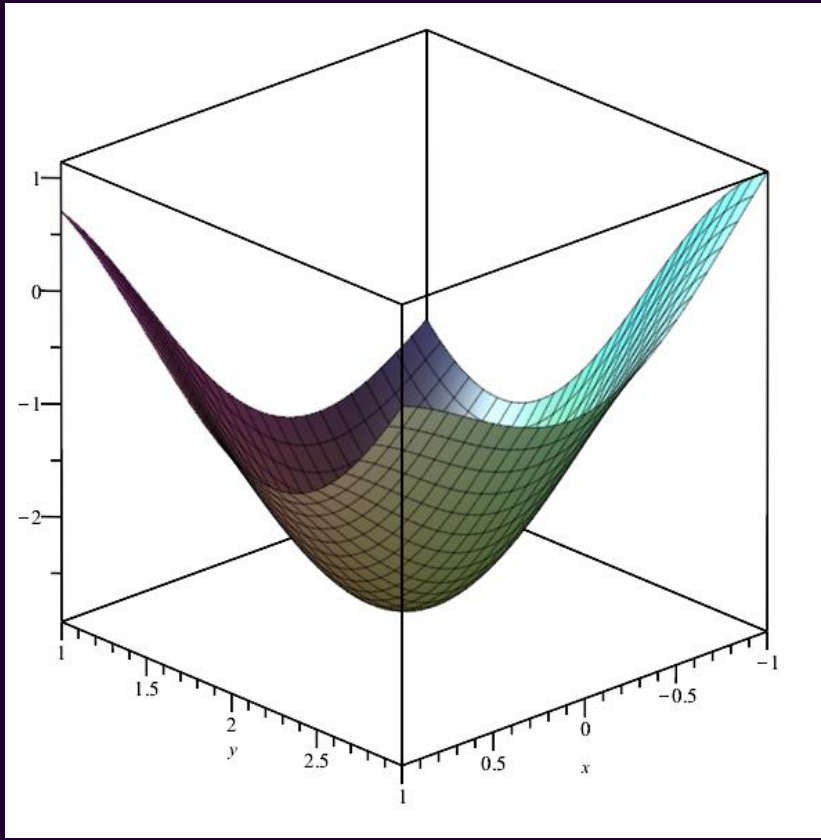


Example

- There is a minimum close to $x = -0.2$ and $y = 1.8$

$$\mathbf{u} = \begin{pmatrix} x \\ y \end{pmatrix}$$

$$\sin(2x + 5) + \sin(y - 3) + \sin(x - 2y - 4)$$



Example

- We can calculate the gradient and Hessian:

$$\mathbf{u} = \begin{pmatrix} x \\ y \end{pmatrix}$$

$$\sin(2x + 5) + \sin(y - 3) + \sin(x - 2y - 4)$$

  - We find these are:

$$\vec{\nabla} f(\mathbf{u}) = \begin{pmatrix} 2\cos(2x+5) + \cos(x-2y-4) \\ \cos(y-3) - 2\cos(x-2y-4) \end{pmatrix}$$

$$\mathbf{J}\left(\vec{\nabla} f\right)(\mathbf{u}) = \begin{pmatrix} -4\sin(2x+5) - \sin(x-2y-4) & 2\sin(x-2y-4) \\ 2\sin(x-2y-4) & -\sin(y-3) - 4\sin(x-2y-4) \end{pmatrix}$$

Example

- Let us start with $\mathbf{u}_0 = \begin{pmatrix} -0.2 \\ 1.8 \end{pmatrix}$ where $f(\mathbf{u}_0) = -2.9242734350$
  - Thus,

$$\vec{\nabla} f(\mathbf{u}_0) = \begin{pmatrix} -0.1703496333 \\ 0.2544469134 \end{pmatrix}$$

$$\mathbf{J}\left(\vec{\nabla} f\right)(\mathbf{u}_0) = \begin{pmatrix} 4.9733073599 & -1.9970866907 \\ -1.9970866907 & 4.9262124675 \end{pmatrix}$$

  - Thus, solving $\mathbf{J}\left(\vec{\nabla} f\right)(\mathbf{u}_0)\Delta\mathbf{u}_0 = -\vec{\nabla} f(\mathbf{u}_0)$ yields $\Delta\mathbf{u}_0 = \begin{pmatrix} 0.0161387791 \\ -0.0451089704 \end{pmatrix}$
  - Thus, $\mathbf{u}_1 \leftarrow \mathbf{u}_0 + \Delta\mathbf{u}_0 = \begin{pmatrix} -0.2 \\ 1.8 \end{pmatrix} + \begin{pmatrix} 0.0161387791 \\ -0.0451089704 \end{pmatrix} = \begin{pmatrix} -0.1838612209 \\ 1.7548910296 \end{pmatrix}$

    and $f(\mathbf{u}_1) = -2.9313971404$

Example

- Continuing with $\mathbf{u}_1 = \begin{pmatrix} -0.1838612209 \\ 1.7548910296 \end{pmatrix}$ where $f(\mathbf{u}_1) = -2.9313971404$
  - Thus,

$$\vec{\nabla} f(\mathbf{u}_1) = \begin{pmatrix} -0.0003992889 \\ 0.0006556248 \end{pmatrix}$$

$$\mathbf{J}\left(\vec{\nabla} f\right)(\mathbf{u}_1) = \begin{pmatrix} 4.9743445043 & -1.9743466419 \\ -1.9743466419 & 4.8961243074 \end{pmatrix}$$

  - Thus, solving $\mathbf{J}\left(\vec{\nabla} f\right)(\mathbf{u}_1)\Delta\mathbf{x}_1 = -\vec{\nabla} f(\mathbf{u}_1)$ yields $\Delta\mathbf{u}_1 = \begin{pmatrix} 0.0000322891 \\ -0.0001208864 \end{pmatrix}$
  - Thus, $\mathbf{u}_2 \leftarrow \mathbf{u}_1 + \Delta\mathbf{u}_1 = \begin{pmatrix} -0.1838289318 \\ 1.7547701432 \end{pmatrix}$

    and $f(\mathbf{u}_2) = -2.9313971865$

Example

- Finally, with $\mathbf{u}_2 = \begin{pmatrix} -0.1838289318 \\ 1.7547701432 \end{pmatrix}$ where $f(\mathbf{u}_2) = -2.9313971865$
  - Thus,

$$\vec{\nabla} f(\mathbf{u}_2) = \begin{pmatrix} -0.0000000057 \\ 0.0000000097 \end{pmatrix}$$

$$\mathbf{J}\left(\vec{\nabla} f\right)(\mathbf{u}_2) = \begin{pmatrix} 4.9743213760 & -1.9742590585 \\ -1.9742590585 & 4.8959878126 \end{pmatrix}$$

  - Thus, solving $\mathbf{J}\left(\vec{\nabla} f\right)(\mathbf{u}_2)\,\Delta\mathbf{u}_2 = -\vec{\nabla} f(\mathbf{u}_2)$ yields $\Delta\mathbf{u}_2 = \begin{pmatrix} 0.0000000004 \\ -0.0000000018 \end{pmatrix}$
  - Thus, $\mathbf{u}_3 \leftarrow \mathbf{u}_2 + \Delta\mathbf{u}_2 = \begin{pmatrix} -0.1838289313 \\ 1.7547701414 \end{pmatrix}$

    and $f(\mathbf{u}_3) = -2.9313971865$

Implementation

- The images were made, and the results were verified in Maple
- The MATLAB code for the previous example was as follows:

```
>>  f = @(u)( sin(2*u(1)+5) + sin(u(2)-3) + sin(u(1)-2*u(2)-4) );
>> df = @(u)( [2*cos(2*u(1)+5) +   cos(u(1)-2*u(2)-4)
                 cos(u(2)-3) - 2*cos(u(1)-2*u(2)-4)] );
>> Jf = @(u)( [-4*sin(2*u(1)+5) - sin(u(1)-2*u(2)-4),              2*sin(u(1)-2*u(2)-4)
                              2*sin(u(1)-2*u(2)-4), -sin(u(2)-3) - 4*sin(u(1)-2*u(2)-4)] );

>> u = [-0.2 1.8]';
>> for i = 1:4
     f(u)
    df(u)
    Jf(u)
    du = Jf(u) \ -df(u)
    u = u + du              # overwrite 'u'
end
```

When to use?

- If you are simulating a system you have modelled,
  chances are you have mathematical descriptions of each of the components in your system
  - Consequently, you can explicitly calculate both the gradient and the Hessian exactly

- If you had a black-box function *f* for which you could not compute the partial derivatives,
  other techniques are preferable

Summary

- Following this topic, you now
  - Are aware Newton's method can be used to find extrema for real-valued functions of vector variables
  - You understand that you must compute both the gradient and the Jacobian of the gradient (the Hessian)
  - Understand that other techniques are preferable if the partial derivatives cannot be precisely
  - Have seen two examples

References

[1] https://en.wikipedia.org/wiki/Newton%27s_method_in_optimization

Acknowledgments

None so far.

Colophon

These slides were prepared using the Cambria typeface. Mathematical equations use Times New Roman, and source code is presented using `Consolas`. Mathematical equations are prepared in MathType by Design Science, Inc.

Examples may be formulated and checked using Maple by Maplesoft, Inc.

The photographs of flowers and a monarch butter appearing on the title slide and accenting the top of each other slide were taken at the Royal Botanical Gardens in October of 2017 by Douglas Wilhelm Harder. Please see

https://www.rbg.ca/

for more information.

Disclaimer

These slides are provided for the ECE 204 *Numerical methods* course taught at the University of Waterloo. The material in it reflects the author's best judgment in light of the information available to them at the time of preparation. Any reliance on these course slides by any party for any other purpose are the responsibility of such parties. The authors accept no responsibility for damages, if any, suffered by any party as a result of decisions made or actions based on these course slides for any other purpose than that for which it was intended.