TANA15/Lecture 6 - Contents

Systems of Non-Linear Equations

- Newtons Method in one dimension: Existance. Convergence Speed. Error Estimate. The Secant Method.
- Fixed Point Iteration. Contraction Mapping Theorem.
- Taylor series expansions. The Jacobian Matrix. Newton's Method.
- Updating methods. Broyden's method.

Applications

- Image inpainting. Soccer Ball trajectory.

Non-Linear Equations

We want to solve an equation,

$$f(x) = 0, \qquad x \in \Omega.$$

Example Consider the equations:

$$f(x) = x^3 - 2 + \mathrm{e}^{-x} = 0,$$

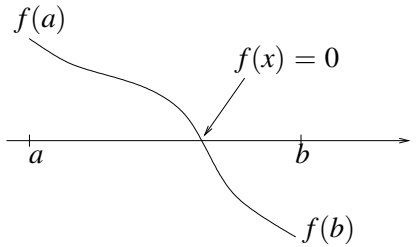
or,

$$f(x) = \begin{pmatrix} x_1^2 - x_2 - 1 \\ -x_1 + x_2^2 - 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Remarks Often $\Omega = \mathbb{R}^n$ and f is a mapping from $\mathbb{R}^n$ to $\mathbb{R}^m$. Existance, Uniqueness, and Stability is much more complicated than for linear systems.

Intermediate Value Theorem

Theorem If $f(x)$ is continuous on $[a,b]$ and $c \in [f(a), f(b)]$ then there is a $x \in [a,b]$ such that $f(x) = c$.



Remark This is a practical existance criteria for roots $f(x) = 0$ in one dimension. For systems of equations the situation is more complex.

Lemma A *fixed point* x^* of the Newton iteration,

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)},$$

is a root of the equation $f(x) = 0$.

Example Solve the equation $f(x) = \mathrm{e}^{-x} - x = 0$

n	x_n	f_n/f_n'
0	1.00000000000000	$4.6 \cdot 10^{-1}$
1	0.53788284273999	$-2.9 \cdot 10^{-2}$
2	0.56698699140541	$-1.6 \cdot 10^{-4}$
3	0.56714328598912	$-4.4 \cdot 10^{-9}$
4	0.56714329040978	$-7.1 \cdot 10^{-17}$

Convergence Speed? Error estimate? Stability?

If it is difficult to compute $f'(x)$ we can approximate

$$f'(x_n) \approx \frac{f(x_n + h) - f(x_n - h)}{2h},$$

or

$$f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}.$$

Remarks The second option is called the *Secant method*. Slower convergence but doesn't require $f'(x)$.

Question How to generalize this to systems of non-linear equations, i.e.

$$f(x) = 0, \qquad f : \mathbb{R}^n \mapsto \mathbb{R}^m?$$

Fixed Point Iteration

Definition Let $g() : \mathbb{R}^n \mapsto \mathbb{R}^n$. A *fixed point* x^* of g satisfies,

$$x^* = g(x^*).$$

Method Pick a starting approximation $x^{(0)}$ and iterate,

$$x^{(k+1)} = g(x^{(k)}), \qquad k = 0, 1, 2 \ldots$$

Existance of a fixed point? When does the iteration converge?

The Contraction Mapping Theorem

Let x^* be a point in $\mathbb{R}^n$, $r > 0$, and let

$$S_r = \{x \,|\, \|x - x^*\| \leq r\}$$

be a ball around x^*.

Definition If there is an $r > 0$ such that

$$\|g(x) - x^*\| \leq |L|\|x - x^*\|, \qquad \text{for all } x \in S_r,$$

where $|L| < 1$, then g is called a *contraction mapping*, and L a *Lipschitz constant*.

Theorem Let g be a contraction mapping. Then the iteration

$$x^{(k+1)} = g(x^{(k)}), \quad x^{(0)} \in S_r$$

converge to a unique fixed point $x^* \in S_r$. Also

$$\|x^{(k)} - x^*\| \leq |L|^k \|x^{(0)} - x^*\|.$$

Lemma Let g be differentiable with fixed point x^*. If $\|g'(x^*)\| < 1$ then $g()$ is a contraction mapping.

Example We want to solve the system of equations

$$f(x) = \begin{pmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \end{pmatrix} = \begin{pmatrix} x_1^2 + x_2^2 - 1 \\ (x_1 - 0.5)^2 + x_2^2 - 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

We use the fixed point iteration,

$$x^{(k+1)} = x^{(k)} - f(x^{(k)}), \quad \text{and } x_0 = (0.4, 1.2)^T.$$

k	$x_1^{(k)}$	$x_2^{(k)}$	$\|x^{(k)} - x^*\|$
0	0.4000	1.2000	$2.76 \cdot 10^{-1}$
1	-0.2000	0.7500	$5.00 \cdot 10^{-1}$
5	0.2661	0.8528	$1.17 \cdot 10^{-1}$
10	0.2034	0.9652	$4.67 \cdot 10^{-2}$
15	0.2581	0.9686	$8.10 \cdot 10^{-3}$
20	0.2486	0.9682	$1.43 \cdot 10^{-3}$

Definition If, asymptotically,

$$\|x^{(k+1)} - x^*\| \leq C\|x^{(k)} - x^*\|, \qquad C < 1,$$

then the convergence rate is *linear*.

Remark The Contraction Mapping theorem can be used to prove existance of a unique solution to $f(x) = 0$ even if we don't use fixed point iteration for solving the equation.

Image Inpainting

Suppose we have an image I with an irritating detail that we want remove. Can we solve the problem?

The part of the image we want to change is displayed. This is an *interpolation problem*.

Definition An *image* $I \in \mathbb{R}^{n \times m}$ is a matrix where each *pixel* I_{ij} has a value that represents a *color*.

Definition The interpolation *mask* $M \in \mathbb{R}^{n \times m}$ has elements $M_{ij} = 1$ if the corresponding *pixel* should be replaced by an interpolated value and $M_{ij} = 0$ otherwise.

Remark An interpolation mask is a very efficient method for describing a complicated geometry.

Harmonic Interpolation

Definition Suppose $I(x,y)$ is defined on a domain in $\mathbb{R}^2$ and unknown in a subdomain Ω. Harmonic interpolation means computing $I_H(x,y)$ as the solution to

$$\begin{cases} \Delta I_H = 0, & \text{in } \Omega, \\ I_H(x,y) = I(x,y), & \text{on } \partial\Omega. \end{cases}$$

Remark This is implemented using finite differences as a linear system of equations $AI_H = b$.

The result after using Harmonic inpainting and the original image.

Using a linear differential operator $\Delta I = 0$ we always get a smooth image. If we want sharp features we need non-linear operators.

Total Variation Inpainting

Definition Let $I(x,y)$ be defined on a domain in $\mathbb{R}^2$, and unknown in the subdomain Ω. The *Total Variation Inpainting* interpolant is

$$\begin{cases} -\nabla \cdot \left(\frac{1}{|\nabla I_{TV}|} \nabla I_{TV} \right) = 0, & \text{in } \Omega, \\ I_{TV}(x,y) = I(x,y), & \text{on } \partial\Omega. \end{cases}$$

Algorithm Pick a starting guess $I^{(0)}$ and iterate

$$G(i,j) = |\nabla I_{TV}^{(k-1)}(i,j)|^{-1}, \quad \nabla \cdot (G \nabla I_{TV}^{(k)}) = 0.$$

Remark This is *fixed point iteration* $I_{TV}^{(k)} = \varphi(I_{TV}^{(k-1)})$. Convergence can be proved.

The results from using the *Total variation principle*. Slightly sharper details inside the interpolation region.

Solving linear systems of equations is likely the most important problem in numerical linear algebra.

Taylor Series

Example Let $f : \mathbb{R}^2 \mapsto \mathbb{R}^2$. Approximate f by its linear part.

Lemma Any, twice differnetiable, function $f(x) : \mathbb{R}^n \mapsto \mathbb{R}^n$ can be written as a Taylor series,

$$f(x) = f(x^{(0)}) + J_f(x^{(0)})(x - x^{(0)}) + \mathcal{O}(\|x - x^{(0)}\|^2),$$

where $J_f \in \mathbb{R}^{n\times n}$ is the *Jacobian matrix* defined by

$$(J_f(x))_{i,j} = \frac{\partial f_i}{\partial x_j}(x).$$

Newtons method

Algorithm Let $x^{(0)} \approx x^*$. Find a root root $f(x^*) = 0$ by

for $k = 1, 2, 3, \ldots$

Solve $J_f(x^{(k)})s^{(k)} = -f(x^{(k)})$.

Update $x^{(k+1)} = x^{(k)} + s^{(k)}$.

if $\|s^{(k)}\| < tol$ **then** stop.

end

Remark If $\bar{x}$ is an approximate solution then the error can be estimated as,

$$\|\bar{x} - x^*\| \approx \|J_f^{-1}(\bar{x})f(\bar{x})\|.$$

Example Solve the system of equations

$$f(x) = \begin{pmatrix} f_1(x_1,x_2) \\ f_2(x_1,x_2) \end{pmatrix} = \begin{pmatrix} x_1^2 + x_2^2 - 1 \\ (x_1 - 0.5)^2 + x_2^2 - 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

using Newtons Method. The Jacoian is given by

$$J_f(x) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 2x_1 & 2x_2 \\ 2(x_1 - 0.5) & 2x_2 \end{pmatrix}.$$

Implement the function and Jacobian in Matlab

```
f=@(x) [x(1)^2+x(2)^2-1;(x(1)-0.5)^2+x(2)^2-1];
J=(x) 2*[ x(1),x(2); x(1)-0.5 , x(2)];
```


Matlab The Newton iteration is implemented as

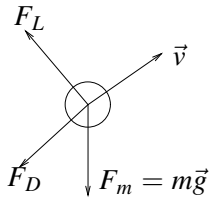
```
>>  x=[0.4 ; 1.2];
>>  for k=1:10, s=-J(x)\f(x); , x=x+s; , end
```

The results are

k	$x_1^{(k)}$	$x_2^{(k)}$	$\|x^{(k)} - x^*\|$
0	0.4000	1.2000	$2.76 \cdot 10^{-1}$
1	0.2500	1.0000	$3.18 \cdot 10^{-2}$
2	0.2500	0.9688	$5.04 \cdot 10^{-4}$
3	0.2500	0.9682	$1.31 \cdot 10^{-7}$
4	0.2500	0.9682	$9.10 \cdot 10^{-15}$
5	0.2500	0.9682	$1.24 \cdot 10^{-16}$

This is clearly quadratic convergence!

Example Trajectory Analysis of a Soccer Ball.



A Soccer ball that travels through air with velocity $\vec{v}$ is subject to a *Drag force* F_D, *gravity pull* F_m, and a *Magnus force* F_L because of its spin.

Question How can we hit the ball so that it bounces off a certain spot on the ground.

The *Drag Force* acts as opposite to the *velocity vector* $\vec{v}$ and can be written

$$F_D = -\frac{1}{2}\rho A C_D |\vec{v}|\vec{v},$$

where ρ is the air density, A is the cross section area of the ball, and C_D is the drag coefficient.

The *Magnus Force* arises when the ball spins while moving through the air. It can be written as

$$F_L = \rho A C_L(\vec{\omega} \times \vec{v}),$$

where ω is the angular momentum and ρ is the density of the air.

Let the *State vector* be

$$S(t) = \begin{pmatrix} \vec{p}(t) \\ \vec{v}(t) \end{pmatrix},$$

then the trajectory of the Soccer ball can be obtained by solving the following ODE

$$S'(t) = \begin{pmatrix} \vec{p}\,'(t) \\ \vec{v}\,'(t) \end{pmatrix} = \begin{pmatrix} \vec{v}(t) \\ (F_D + F_m + F_L)/m \end{pmatrix},$$

Given an initial position $\vec{p}(0)$, an initial velocity $\vec{v}(0)$, and the angular momentum $\vec{\omega}$ we can compute the full trajectory of the soccer ball!

Matlab The spot where the ball first bounce is a function of the initial velocity $\vec{v}_0$ and initial position $\vec{p}_0$.

```
function [ Pe , Ve ] = FindFinalPosition( P0 , V0 )
   S0=[P0;V0];

   % Add the event that the ball hits the ground
   options = odeset('Events',@HitGround,'RelTol',10^-9);

   % Solve the system of ODEs
   [t,S]=ode45(@SoccerODE,[0:0.1:100],S0,options);

   % Collect position and velocity at the final time.
   Pe=S(end,1:3)';
   Ve=S(end,4:6)';
```

This is a well defined function but can't easily compute its Jacobian matrix.

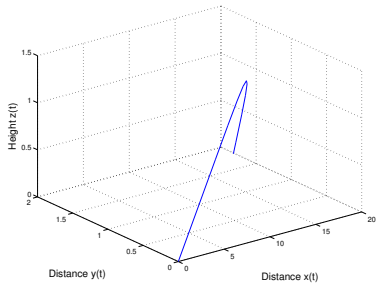
The Jacobian can be computed numerically. Pick a small $h > 0$ and approximate

$$(J_f(x))_{i,j} = \frac{\partial f_i}{\partial x_j}(x) \approx \frac{f_i(x + h\vec{e}_j) - f_i(x - h\vec{e}_j)}{2h}.$$

Requires $2n$ additional function evaluations. The matrix J_f is computed one column at a time.

```
function [ J] = BallJacobian(P0 , V0  )
  h=10^-4;
  J=zeros(3,3); I=eye(3);
  for j=1:3
    tmp=FindFinalPosition( P0 ,V0+I(:,j)*h );
    tmp= tmp-FindFinalPosition( P0 ,V0-I(:,j)*h );
    tmp=tmp/2/h;
    J(:,j)=tmp(1:3);
 end
```


Example Hit the Soccer ball from $p_0 = (0,0,0)^T$ with initial velocity $v_0 = (25,0,6)^T$ and angular momentum $\omega = (0, 2\pi, 12\pi)^T$.



The ball bounces at $p_e = (19.88, 1.81, 0.00)^T$. We want the ball to bounce at $p^* = (23,5,0)$ instead. What initial velocity to use?

The Newton method for solving $f(v_0) - p_e = 0$ is implemented as

```
Vk=[25 0 6]';Pe=[23,5,0]';
for k=1:10
   Fk = FindFinalPosition( P0 , Vk ) - Pe;
   Jk = BallJacobian(P0 , Vk);
   sk = -Jk\Fk;
   Vk=Vk+sk;
 end;
```

Remark Stopping criteria based on $\|S_k\|_2$ works well. How accurate are F_k and J_k here?

k	$v_x^{(k)}$	$v_y^{(k)}$	$v_z^{(k)}$	$\|p_e^{(k)} - p^*\|$
1	25.0000	0.0000	6.0000	$4.46 \cdot 10^{0}$
2	19.5699	2.6733	8.6637	$1.95 \cdot 10^{0}$
3	20.2549	1.5274	9.3487	$5.45 \cdot 10^{-2}$
4	20.2788	1.5736	9.3485	$3.40 \cdot 10^{-5}$
5	20.2792	1.5737	9.3483	$5.62 \cdot 10^{-9}$
6	20.2792	1.5737	9.3483	$4.19 \cdot 10^{-14}$
7	20.2792	1.5737	9.3483	$1.62 \cdot 10^{-13}$
8	20.2794	1.5737	9.3482	$1.90 \cdot 10^{-9}$
9	20.2795	1.5738	9.3481	$1.26 \cdot 10^{-9}$
10	20.2805	1.5740	9.3477	$3.88 \cdot 10^{-8}$

Remarks Quadratic convergence after iteration $k = 2$. Good starting guess is *very important*. Tricky to get Newtons method to converge. Can we avoid computing the Jacobian and still get good convergence?

Secant Updating Methods

Let $f(x) : \mathbb{R}^n \mapsto \mathbb{R}^n$ and we attempt to solve $f(x) = 0$. The Newton method can be written

$$x^{(k+1)} = x^{(k)} - J_f^{-1}(x^{(k)})f(x^{(k)}), \qquad k = 0, 1, 2, \ldots,$$

where $J_f(x^{(0)})$ is the *Jacobian matrix* and $x^{(0)}$ is the starting guess.

Remarks The Newton method works *extremely well* but it is not always easy to compute the Jacobian matrix. In one dimension *the secant method* address this. For vector valued functions

$$\frac{\partial f}{\partial \vec{v}}(x^{(k)}) = J_f(x^{(k)})(x^{(k)} - x^{(k-1)}) \approx \frac{f(x^{(k)}) - f(x^{(k-1)})}{\|x^{(k)} - x^{(k-1)}\|_2}$$

One directional derivative isn't enough to determine J_f. Alternative?

Lemma Suppose $B_k \approx J_f(x^{(k)})$, $s^{(k)} = x^{(k+1)} - x^{(k)}$, and $y^{(k)} = f(x^{(k+1)}) - f(x^{(k)})$. Then uv^T, where

$$u = \frac{y^{(k)} - B_k s^{(k)}}{\|s^{(k)}\|_2^2}, \quad \text{and,} \quad v = s^{(k)},$$

is the smallest rank one matrix such that

$$B_{k+1}s^{(k)} = (B_k + uv^T)s^{(k)} = y^{(k)}.$$

Remark The new approximate Jacobian $B_{k+1} \approx J_f(x^{(k+1)})$ is constructed so the directional derivative in the direction $-s^{(k)}$ agrees with a finite difference approximation.

Broyden's Method

Algorithm Given $x^{(0)} \approx x^*$ and $B_0 \approx J_f(x^{(0)})$ do,

1. **for** $k = 1, 2, 3, \ldots$

   Solve $B_k s^{(k)} = -f(x^{(k)})$.

   Update $x^{(k+1)} = x^{(k)} + s^{(k)}$.

   Update $B_{k+1} = B_k + \frac{1}{\|s^{(k)}\|_2^2}(f(x^{(k+1)}) - f(x^{(k)}) - B_k s^{(k)})(s^{(k)})^T$.

   if $\|s^{(k)}\| < tol$ **then** stop.

3. **end**

Remark Initial Jacobian B_0 can be computed numerically; but often $B_0 = I$ is used.

Example Soccer Ball Trajectory problem: Iteration sequence $\{x_k\}$ using Broyden's method and $B_0 = I$.

k	$v_x^{(k)}$	$v_y^{(k)}$	$v_z^{(k)}$	$\|p_e^{(k)} - p^*\|$
1	25.0000	0.0000	6.0000	$1.63 \cdot 10^{0}$
2	26.1167	1.1910	6.0000	$6.40 \cdot 10^{-1}$
3	27.0406	1.5283	6.0000	$1.51 \cdot 10^{-1}$
4	27.2332	1.4682	6.0000	$3.74 \cdot 10^{-2}$
5	27.2883	1.4434	6.0000	$1.81 \cdot 10^{-3}$
6	27.2910	1.4421	6.0000	$2.24 \cdot 10^{-5}$
7	27.2910	1.4421	6.0000	$4.76 \cdot 10^{-8}$
8	27.2910	1.4421	6.0000	$1.05 \cdot 10^{-10}$
9	27.2910	1.4421	6.0000	$7.24 \cdot 10^{-14}$

Remark Similar convergence rate as the Newton method. Only one function evaluation $f(x^{(k)})$ at each step.