APM 523
Ch. 11, Nonlinear Optimization

Matt Kinsinger, Hongjun Choi, Adarsh Akkshai

October 13, 2017

Nonlinear optimization is concerned with optimization of variables involved to form a solution, such that they satisfy the set of subject constraints or n equalities. Rather than optimizing the objective function as in linear optimization. More formally, if a vector $r \in \mathbb{R}^n$ such that:

$$r(x) = \begin{bmatrix} r_1(x) \\ r_2(x) \\ r_3(x) \\ \vdots \\ r_n(x) \end{bmatrix}$$

We assume that $r_i : \mathbb{R}^n \to \mathbb{R}$, i = 1, 2, ..., n is smooth. A vector x^* that satisfies $r(x^*) = 0$ is a *solution* or *root* of the nonlinear equations.

Preliminary Remarks

1) Heart of all optimization techniques is achieved by Newton's method, Quasi-Newton methods are generally less useful in nonlinear equations.

2) The number of equations is equal to the number of variables, unlike nonlinear least squares.

3) Quadratic convergence proofs for nonlinear optimization techniques involve only first order differentials, while unconstrained and least squares require second order differentials.

4) The system can have multiple solutions, in linear optimization this can be viewed as local minima or maxima. However, in nonlinear equations all set of solutions will result in the same objective (if one were to think of it as a linear model). Example: $r(x) = \sin(5x) - x$, has 3 unique solutions which all satisfy the mathematical equality of $r(x^*) = 0$, yet these may hold no physical meaning for the observer.

5) The vector function r is continuously differentiable in the region $\mathbb{D}$ containing the values of x we are interested in. ($D \subset \mathbb{R}^n$)

   * The Jacobian $\mathcal{J}(x)$ exists and is continuous in $\mathbb{D}$

7) x^* satisfying $r(x^*) = 0$ is a *degenerate solution* if $\mathcal{J}(x^*)$ is singular, and *nondegenerate solution* otherwise.

1 Local Algorithms

1.1 Newtons Method for Nonlinear Equations

Originally Newton's method for minimizing $f : \mathbb{R}^n \to \mathbb{R}$ is done my minimizing the Taylor series

$$f(x_k + p_k) = f(x_k) + \nabla f(x_k + t_k p_k)^T p_k, \quad \text{where} \quad p \in \mathbb{R}^n$$

with the Newton step.

Similarly, for nonlinear equations we extend this to form Theorem 1.

Theorem 1:
Suppose $r : \mathbb{R}^n \to \mathbb{R}^n$ and $x, p \in \mathbb{D}$ and r is continuously differentiable in $\mathbb{D}$

$$r(x_k + p_k) = r(x_k) + \int_0^1 \mathcal{J}(x_k + t_k p_k) p_k dt \tag{1}$$

where, $x_k = [x_{1,k} \dots x_{n,k}]^T$, is the k^{th} iterate of the solution $x_k \to x^*$ for mapping $r_i : \mathbb{R}^n \to \mathbb{R}$

We then define a linear model $M_k(p)$ of $r(x_k + p_k)$ by approximating the second term in (1), i.e.

$$M_k(p) \triangleq r(x_k) + \mathcal{J}(x_k)p \tag{2}$$

Where, Newton's step is solved exactly as $M_k(p) = 0$, solving for p, we get $p = -\mathcal{J}(x_k)^{-1} r(x_k)$

Algorithm 1 Newton's Method for Nonlinear Equations

```
Choose x_0
for k = 0, 1, 2, ... do
    J(x_k) p_k = -r(x_k)
    x_{k+1} ← x_k + p_k
end(for)
```

Additional emphasis is given on linear models rather than quadratic ones because of rapid convergence and normally linear equations have a definite solution.

We now analyze the convergence of this method.

Theorem 2

Suppose a r is continuously differentiable function in the convex set $\mathbb{D} \subset \mathbb{R}^n$. Let $x^* \in \mathbb{D}$ be a non degenerate solution of $r(x) = 0$, and let $\{x_k\}$ be the sequence of iterates generated by Algorithm 1. Then as $x_k \to x^*$, when x_k is close to x^* we have:

$$\left\{ \begin{array}{ll} x_{k+1} - x^* = o(\|x_k - x^*\|), & \text{local Q-Superlinear Convergence} \\ & \text{If r is continuously differentiable} \\ x_{k+1} - x^* = O(\|x_k - x^*\|^2), & \text{local Q-quadratic convergence} \\ & \text{r is Lipschitz continuously differentiable} \end{array} \right\} \tag{3}$$

Where Lipschitz continuous means the Jacobian $\mathcal{J}$ of r , should have a $\beta_L \neq 0$, such that:

$$\|\mathcal{J}(x_0) - \mathcal{J}(x_1)\| \leq \beta_L \|x_0 - x_1\|$$

1.1.1 Important takeaways

- The convergence is superlinear/quadratic only when x_k and $x^* \in \mathcal{D}$
- More briefly, if the starting point is remote from the solution then $\mathcal{J}(x_k)$ may be singular.
- Finding $\mathcal{J}$ may not always be easy.
- If n is too large, Newton step p_k might take longer to converge, just as line search.
- The worst case is when x^* turns $\mathcal{J}(x^*)$ singular.

1.2 Inexact Newton methods

To combat the singularity and the approach of solving for $r(x_k)$ exactly might be hard through (2). The equivalent inexact Newton's method was extended to nonlinear equations. To find an inexact solution is to satisfy;

$$\|r_k + \mathcal{J}_k p_k\| \leq \eta_k \|r_k\|, \quad \text{for some} \quad \eta_k \in [0, \eta] \tag{4}$$

where $\eta \in [0, 1)$ is a constant, $\{\eta_k\}$ is called the forcing sequence. The general framework for inexact methods is:

First we shall discuss the convergence of this method and later discuss it's usability.

Theorem 3

Suppose r is continuously differentiable in the convex open set $\mathbb{D}$ ($\mathbb{D} \subset \mathbb{R}^n$). Then here x^* is the nondegenerate solution of $r(x) = 0$ and $\{x_k\}$ be the sequence of iterates from Algorithm 2. Then when x_k is close to x^*:

1) If η in 4 is sufficiently small, the convergence of $\{x_k\}$ to x^* is Q-linear.

Algorithm 2 Ineact Newton for Nonlinear Equations

1: Given $\eta \in [0, 1)$;
2: Choose x_0
3: **for** $k = 0, 1, 2 \ldots$ **do**
4: Choose forcing parameter $\eta_k \in [0, \eta]$
5: Find a vector p_k that satisfies 4
6: $x_{k+1} \leftarrow x_k + p_k$
7: **end(for)**

2) If $\eta_k \to 0$, the convergence is Q-superlinear.

3) If, in addition, $\mathcal{J}(.)$ is Lipschitz continuous in a neighbhourhood of x^* and $\eta_k = O(\|r_k\|)$, convergence is Q-quadratic.

1.2.1 Important Takeaways

1) Convergence is subject only to condition 4 and not a particular method to calculate p_k

2) Some class of iterative techniques involve in general use a system of equations such that $\mathcal{J}.p = -r$ where d is used in place of p for every iteration of the solution and requires storage. The tradeoff is better more accurate convergence.

   $\mathcal{J}(.)$ need not be explicitly calculated to measure $\mathcal{J}d$ (finite difference approach can be used or automatic differentiation yield faster results)

1.3 Broyden's Method

This class of methods are also known to be secant methods or Quasi-Newton methods. The advantage of this class is that computational of $\mathcal{J}(x_k)$ is not required. They have self updating Jacobians, almost close to the true Jacobian (B_k). Analogous to the original model of Newton's method, 2, here we have

$$M_k(p) = r(x_k) + B_k p \tag{5}$$

Setting the model equal to zero for the solution of p_k, provided B_k is nonsingular we have $p_k = -B_k^{-1} r(x_k)$

The most practical algorithm for updating B_k is:

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k) {s_k}^T}{s_k^T s_k} \tag{6}$$

Where B_k satisfies the secant equation of $y_k = B_{k+1} s_k$, $s_k = x_{k+1} - x_k$, $y_k = r(x_{k+1}) - r(x_k)$

The convergence of Broyden's method is superlinear (under certain assumptions)

Algorithm 3 Broyden's Method

1: Choose x_0 and a nonsingular initial Jacobian approximation B_0
2: **for** $k = 0, 1, 2 \ldots$ **do**
3: Calculate a solution p_k for $B_k p_k = -r(x_k)$
4: Choose α_k by performing **LineSearch** on p_k
5: $x_{k+1} \leftarrow x_k + \alpha_k p_k$
6: $s_k \leftarrow x_{k+1} - x_k$
7: $y_k \leftarrow r(x_{k+1}) - r(x_k)$
8: Obtain $B_{k=1}$ from 6
9: **end(for)**

Theorem 4
Suppose the assumptions of Theorem 2 hold. Then there are positive constants κ and δ such that if the starting point x_0 and starting approximate B_0 satisfy $\|x_0 - x^*\| \leq \delta$ and $\|B_0 - \mathcal{J}(x^*)\| \leq \kappa$ the sequence $\{x_k\}$ generated by Broyden's method is well defined and converges Q-superlinearly to x^*

1.3.1 Important Takeaways

- Newton's method clearly is faster than Broyden's method(except towards the end). However, the functional error $\|r(x_k)\|_2$ is much lower than Newton.
- The initial approximate of B_0 plays an important role, some recommendation is to choose it to be $\mathcal{J}(x_0)$. But what is $\mathcal{J}(x_0)$ is singular, will be stuck.
- B_k is usually dense, hence it will be expensive when n is large. Memory less techniques are possible.

1.4 Tensor Methods

Tensor methods in general augment an extra term to $M_k(p)$ by aiming to capture higher-order behaviour of r.

$$\hat{M_k}(p) = r(x_k) + \mathcal{J}(x_k)p + \frac{1}{2}T_k pp \tag{7}$$

where T_k is a tensor defined by n^3 elements $(T_k)_{ijl}$ whose action on a pair of arbitary vectors u and $v \in \mathbb{R}^n$ is defined by:

$$(T_k uv)_i = \sum_{j=1}^{n}\sum_{l=1}^{n}(T_k)_{ijl}u_j v_l \tag{8}$$

Following Newton's method we can build the second order derivates of r such that

$$(T_k)_{ijl} = [\nabla^2 r_i(x_k)]_{jl}$$

1.4.1 Important Takeaways

- If the true tensor is stored, it would require n times the memory of Newton's method or about $n^3/2$
- For practical purposes, T_k is chosen such that $\hat{M}_k(p)$ interpolates $r(x_k+p)$ by some previous iterates of the algorithm. The number of interpolating points can be modest and usually less than $\sqrt{n}$. The choice of the type of interpolation, their lengths and directions are beyond the scope of this report.
- Similar to Broyden's idea of interpolating the approximate B_k

2 Practical Methods

Practical variants of the Newton-like methods in which line-search and trust-region modifications are made in order to ensure better global convergence behaviour

2.1 Merit Functions

Both Newton's and Broyden's method may require initial starting point near to the solution of $r(x) = 0$ to guarantee convergence. Also, we may experience *cycling* in which the iterates move between distinct regions of the parameter space without approacing a root.

merit function- A scalar valued function of x which indicates whether a new iterate is better or worse than the previous.

In unconstrained optimization problems, the objective function works as a merit function where we say that lower values of the objective are better than higher values. In nonlinear equations, we combine the n components of the vector r in some meaningful way. The most widely used merit function is:

$$f(x) = \frac{1}{2}\|r(x)\|^2 = \frac{1}{2}\sum_{i=1}^{n} r_i^2(x). \tag{9}$$

Note that any root x^* of r has $f(x^*) = 0$, and since $f(x) \geq 0$, all x, each root is a minimizer of f. However, there may be local minimizers y of f where $f(y) > 0$, i.e. y is not a root of r. In these instances we see that

$$\nabla f(y) = J(y)^T r(y) = 0 \tag{10}$$

but $r(y) \neq 0$, thus $J(y)$ is nonsingular (see section **11.3 Continuation/Homotopy Methods** for one technique for avoiding this problem).

2.2 Line search methods

We can obtain algorithms with global convergence properties by applying the line search approach to the equation (2), merit function $f(x) = \frac{1}{2}\|r(x)\|^2$. When it is well defined, the Newton step

$$J(x_k)p_k = -r(x_k) \tag{11}$$

is a descent direction for $f(\cdot)$ whenever $r_k \neq 0$ as below equation (5). Step length α_k are chosen by one of the procedures of Chapter 3 and the iterated are defined by the formula equation (6).

$$p_k^T \nabla f(x_k) = -p_k^T J_k^T r_k = \|r_k\|^2 < 0 \tag{12}$$

$$x_{k+1} = x_k + \alpha p_k, \quad k = 0, 1, 2... \tag{13}$$

For the case of line search that choose α_k to satisfy the Wolfe conditions. We now investigate the values of $cos\theta_k$ for the directions generated by the Newton and inexact Newton iterations. From Theorem 3.2 and equation (5), we can obtain a following bound by combining bounds exact and inexact's Newton step.

$$cos\theta_k = -\frac{p_k^T \nabla f_k}{\|p_k\|\|\nabla f_k\|} \geq \frac{1-\eta^2}{2\|J_k\|\|\nabla J_k^- 1\|(1+\eta)} \geq \frac{1-\eta}{2\kappa(J_k)}. \tag{14}$$

When $\kappa(J_k)$ is large, this lower bound is close to zero and use of the Newton direction may cause poor performance of the algorithm, which highlights a fundamental weakness of the line-search approach. Hence, to prevent this undesirable behavior, we may have to modify the Newton direction as below.

$$p_k = -(J_k^T J_k + \lambda_k I)^{-1} J_k^T r_k. \tag{15}$$

For any $\lambda_k > 0$ the matrix in parentheses is nonsingular, and if λ_k is bounded away from zero, a bounded condition is satisfied. This approach is analogous to the classical **LM**(Levenberg-Marquart) algorithm in Chapter 10. However, the drawback of **LM** approach is that it is difficult to choose λ_k If too large, we can destroy the fast rate of convergence of Newton's method. If λ_k is too small, the algorithm can be inefficient in the presence of Jacobian singularities. A more satisfactory approch is to follow the `trust-region` approach. These two methods can be described as below algorithm tables.

Trust Region Methods

The most widely used trust-region methods for nonlinear equations simply apply Algorithm 4.1 to the merit function $f(x) = \frac{1}{2}\|r(x)\|_2^2$, using $B_k =$

Algorithm 4 Line Search Newton-like Method

1: Given c_1, c_2 with $0 < c_1 < c_2 < \frac{1}{2}$;
2: Choose x_0;
3: **for** $k = 0, 1, 2, ...$
4: Calculate a Newton-like step from equation (4) (regularizing with equation (8) if J_k appears to be near-singular), or $\|r_k + J_k p_k\| \leq \eta \|r_k\|, \quad \eta_k \in [0, \eta]$, $p_k = -B_k^{-1} r(x_k)$, inexact Newton method.
5: **if** $\alpha = 1$ **then** satisfies the Wolfe conditions
6: Set $\alpha_k = 1$;
7: **else**
8: Perform a line search to find $\alpha \geq 0$ that satisfies the Wolfe conditions; **end (if)**
9: $x_{k+1} = x_k + \alpha p_k$
end (for)

$J(x_k)^T J(x_k)$ as the approximate Hessian in the model function $m_k(p)$, which is defined as follows:

$$m_k(p) = \frac{1}{2}\|r_k + J_k p\|_2^2 = f_k + p^T J_k^T r_k + \frac{1}{2} p^T J_k^T p_k. \tag{16}$$

The step p_k is generated by finding an approximate solution of the subproblem.

$$\min_p m_k(p), \quad s.t \quad \|p\| \leq \Delta_k, \tag{17}$$

where Δ_k is the radius of the trust region. The ration ρ_k of actual to predicted reduction, which plays a critical in many trust-region algorithms, is therefore

$$\rho_k(p) = \frac{\|r(x_k)\|^2 - \|r(x_k + p_k)\|^2}{\|r(x_k)\|^2 - \|r(x_k) + J(x_k) p_k\|^2}. \tag{18}$$

We can state the trust-region framework that results from this models as follows.

Dogleg

The dogleg method is a special case of the trust-region algorithm that constructs an approximate solution equation (10) based on the Cauchy point p_k^c and the unconstrained minimizer of m_k. The Cauchy point is

$$p_k^c = -\tau_k (\Delta_k / \|J_K^T r_k\|) J_k^T r_k, \tag{19}$$

where

$$\tau_k = min(1, \|J_K^T r_k\|^3 / (\Delta_k r_k^T J_k (J_k^T J_k) J_k^T)); \tag{20}$$

Additionally, the unconstrained minimizer of $m_k(p)$ is unique when J_k is non-singular. In this case, we denote it by p_k^J and write

$$p_k^J = -(J_k^T J_k)^{-1} (J_k^T r_k) = -J_k^{-1} r_k. \tag{21}$$

The selection of p_k in the dogleg method proceeds as follows.

Algorithm 5 Trust-Region Method for Nonlinear Equations

Given $\overline{\Delta} > 0$, $\Delta_0 \in (0, \overline{\Delta})$, and $\eta \in [0, \frac{1}{4})$;

for $k = 0, 1, 2, ...$
Calculate a p_k as an (approximate) solution of equation (10);
Evaluate ρ_k from equation (11);
if $\rho_k < \frac{1}{4}$ **then**
    $\rho_{k+1} = \frac{1}{4}\|p_k\|$;
else
    if $\rho < \frac{3}{4}$ and $\|p_k\| = \Delta_k$ **then**
        $\rho_{k+1} = \min(2\Delta_k, \overline{\Delta})$
    else
        $\Delta_{k+1} = \Delta_k$
    end (if)
end (if)
if $\rho_k > \eta$ **then**
    $x_{k+1} = x_k + p_k$;
else
    $x_{k+1} = x_k$;
end (if)
end (for)

Algorithm 6 Dogleg

Calculate p_k^c;
if $\|p_k^c = \Delta_k\|$ **then**
    $p_k \leftarrow p_k^c$;
else
    Calculate p_k^J;
    $p_k \leftarrow p_c^J + \tau(p_k^J - p_k^c)$, where τ is the largest value in [0,1]
    such that $\|p_k \leq \Delta_k\|$;
end (if)

3 Continuation/Homotopy Methods

All Newton-based methods may converge to a local minimum of the merit function that is not a solution of the nonlinear system if $J(x)$ is singular in the region of interest. Continuation methods are more likely to converge to a solution of $r(x) = 0$ in difficult cases. We setup an "easy" system of equations for which the solution is obvious. Then we gradually transform the easy system into the original system $r(x)$. We can do this by defining the *homotopy map*:

$$H(x, \lambda) = \lambda r(x) + (1 - \lambda)(x - a), \qquad \lambda \in \mathbb{R},\ a \in \mathbb{R}^n.$$

When $\lambda = 0$ we have the easy problem $0 = H(x, \lambda) = x - a$ with solution $x = a$. When $\lambda = 1$ we have the original system $r(x) = 0$. Intuitively we perform iterations beginning at $\lambda = 0$, $x = a$, and then we gradually increase λ until it is equal to 1 and at each iteration we find the solution x to $H(x, \lambda) = 0$. The value of x when $\lambda = 1$ is our solution to $r(x) = 0$. This process can be plotted as a sequnce of points (λ, x) which create a line called the *zero path*. In some cases each value of λ corresponds to a unique x and we can say that we have x as a function of λ. Thus we can increase λ at each iteration without concern, but it is possible that the *zero path* is not a function $x(\lambda)$ and we have *turning points* in which we cannot continue to follow the *zero path* unless we allow λ to decrease at these turning points.

Practical Continuation Methods

We model the *zero path* parametrically by allowing x and λ to be functions of an independent variable s. Since this is the *zero path* we have:

$$H(x(s), \lambda(s)) = 0, \qquad s \geq 0$$

Taking the total derivative we have

$$\frac{\partial}{\partial x} H(x, \lambda)\dot{x} + \frac{\partial}{\partial \lambda} H(x, \lambda)\dot{\lambda} = 0, \qquad (\dot{x}, \dot{\lambda}) = \left(\frac{dx}{ds}, \frac{d\lambda}{ds}\right).$$

Where $(\dot{x}, \dot{\lambda})$ is the tangent vector to the zero path, and is in the null space of the $n \times (n + 1)$ matrix

$$A = \left[\frac{\partial}{\partial x} H(x, \lambda) \ \ \frac{\partial}{\partial \lambda} H(x, \lambda)\right].$$

If the matrix has full rank, then its null space has dimension one and we can solve for its null space using the QR factorization

$$Q^T A P = [Rw].$$

where P is a permutation matrix, Q is orthogonal, and R is upper triangular. Set

$$v = P \begin{bmatrix} R^{-1}w \\ -1 \end{bmatrix}$$

Then

$$\begin{aligned} Av &= Q\,[R\ \ w]\,Pv \\ &= Q\,[R\ \ w]\,PP\begin{bmatrix} R^{-1}w \\ -1 \end{bmatrix} \\ &= Q\,[R\ \ w]\begin{bmatrix} R^{-1}w \\ -1 \end{bmatrix} \\ &= Q\left[RR^{-1}w - w\right] \\ &= Q \cdot zero \\ &= zero. \end{aligned}$$

Thus we set

$$\left(\dot{x}, \dot{\lambda}\right) = \pm\frac{v}{\|v\|_2}$$

where $\pm$ is chosen to ensure that the direction is towards increasing s. We can terminate this algorithm when we find a value of s for which $\lambda(s) = 1$.

Another approach is an algebraic one. Given a point (x, λ) we again solve for the tangent vector $(\dot{x}, \dot{\lambda})$, then for a small value ϵ we have

$$(x^p, \lambda^p) = (x, \lambda) + \epsilon(\dot{x}, \dot{\lambda})$$

We then apply "corrector" iterations to bring us from (x^p, λ^p) back to a point (x^+, λ^+) along the zero path. In these correction steps we choose the component of (x^p, λ^p) that changed the most over the predictor step and hold it fixed, say that the i^{th} component is held fixed. Then the correction steps take the form

$$\begin{bmatrix} \frac{\partial H}{\partial x} & & \frac{\partial H}{\partial \lambda} \\ & e_i^T & \end{bmatrix}\begin{bmatrix} \delta x \\ \delta\lambda \end{bmatrix} = \begin{bmatrix} -H \\ 0 \end{bmatrix},$$

Where $\frac{\partial H}{\partial x}$, $\frac{\partial H}{\partial \lambda}$, and H are evaluated at the previous iteration of the corrector step so as to work towards undoing the deviation of (x^p, λ^p) from the zero path.

Each of these two Continuation methods relies on the matrix

$$\left[\frac{\partial}{\partial x}H(x,\lambda)\ \ \frac{\partial}{\partial \lambda}H(x,\lambda)\right] \qquad (1)$$

having full rank. This can be guaranteed under certain assumptions.

Theorem 11.9
Suppose that f is twice continuously differentiable. Then for almost all vectors $a \in \mathbb{R}^n$, there is a zero path emanating from $(a, 0)$ along which the $n \times (n+1)$ matrix (1) has full rank. If this path is bounded for $\lambda \in [0, 1)$, then it has an accumulation point $(\overline{x}, 1)$ such that $r(\overline{x}) = 0$. Furthermore, if the Jacobian $J(\overline{x})$ is nonsingular, the zero path between $(a, 0)$ and $(\overline{x}, 1)$ has finite arc length.