Newton and Quasi-Newton Methods

[*]Report 1 on the course "Numerical Optimization".

1st Chen Yihang
Peking University
1700010780

Abstract

In this report, we use vanilla Newton, damped Newton, mixed Newton, Newton-LM method, SR1 and Broyden class (including DFP, BFGS) method to solve the unconstrained nonlinear optimization problem. Polynomial interpolation method is used in the inexact linesearch algorithm.

The main obstacle is the overflow error caused by numerical instability. We adjust a key variable, i.e. the length of each direction before performing line search. Such strategy can be empirically verified to produce good results. A comparision of algorithm is also proposed. For the given initial point, vanilla Newton seems to be impressively well. For very strange functions, Broyden types methods outperforms SR1 method, and BFGS method seems to be the optimal choice. Besides, due to numerical instability, training longer does not necessary means better performace. In practice, we need to find a suitable stopping criterion for each problem.

In the report, it seems that Hessian-free methods does not outperform Hessian-based method in the least square problems. However, we explicitly calculated the Hessian for the least square problems, and the scale of such problem is not large (up to 50). For the minimal surface problem, the number of variable can be around $40^2 = 1600$, and the Hessian is hard to be derived by explicitly. (It lacks the structure for backpropagtion) Hence, in such problem, the most natural way is to use quasi-Newton method.

All the results can be reproduced by executing "main.m", which takes "choice" = 1,2,3 as input, specifying three different test functions.

CONTENTS

I. NEWTON-TYPE METHODS

Given a smooth function $f \in \mathbf{R}^n \to \mathbf{R}$, we use a series of Newton-type methods to solve the unconstrained optimization problem

$$\min_x f(x) \tag{I.1}$$

A. Newton methods

We summarize vanilla Newton, damped Newton, mixed Newton, Newton-LM method below, which are implemented in respective MATLAB scripts.

We use built-in method "cond" (condition number) instead of "det" (determinant) to find out whether a matrix is singular, since "det" is not scale-invariant. Besides, we also use built-in method "chol" (Cholesky decomposition) to find out whether a matrix is positive definite, since its input can only be positive definite matrices.

Since the scale of our problem is not formidable, we do not use Hessian-free method to solve linear program $Gd = g$. Instead, we directly call built-in linear equation solver.

An examplary usage will be

```
try
    chol(Gk);
catch ME
    error('Hessian is not positive definite')
end
```

Some auxiliary quantities, such as number of iterations, number of function evaluations (w.r.t. $f, \nabla f, \nabla^2 f$) is updated in the process. We also record $f(x^\star)$ and $\|\nabla f(x^\star)\|$, where $x^\star$ is the output of the algorithm.

Algorithm 1 Newton method: minimize f, initial point x_0.

Require: $\nabla^2 f$ is positive definite along the trajectory.

procedure NEWTON_VANILLA(f, x_0)
  $x \leftarrow x_0$
  while not converged **do**
    $G \leftarrow \nabla^2 f(x)$
    $g \leftarrow \nabla f(x)$
    $d \leftarrow -G^{-1} g$
    $x \leftarrow x + d$
  end while
  return x
end procedure

Algorithm 2 Damped Newton method: minimize f, initial point x_0.

Require: $\nabla^2 f$ is positive definite along the trajectory.

procedure NEWTON_DAMPED(f, x_0)
  $x \leftarrow x_0$
  while not converged **do**
    $G \leftarrow \nabla^2 f(x)$
    $g \leftarrow \nabla f(x)$
    $d \leftarrow -G^{-1} g$
    $\alpha \leftarrow$ LINESEARCH(f, x, d)
    $x \leftarrow x + \alpha d$
  end while
  return x
end procedure

Algorithm 3 Mixed Newton method: minimize f, initial point x_0.

procedure NEWTON_MIXED(f, x_0)
  $x \leftarrow x_0$
  while not converged **do**
    $G \leftarrow \nabla^2 f(x)$
    $g \leftarrow \nabla f(x)$
    if G is not singular **then**
      $d \leftarrow -G^{-1} g$
      if $g^\top d > \epsilon_1 \|g\| \|d\|$ **then** ▷ non-decreasing
        $d \leftarrow -d$
      end if
      if $|g^\top d| \leq \epsilon_2 \|g\| \|d\|$ **then** ▷ orthogonality
        $d \leftarrow -g$
      end if
    else
      $d \leftarrow -g$
    end if
    $\alpha \leftarrow$ LINESEARCH(f, x, d)
    $x \leftarrow x + \alpha d$
  end while
  return x
end procedure

B. Broyden class method

We first gives an overview of the quasi-Newton method

If we denote the trajectory as $x_k,\ k \geq 0$. For $k \geq 0$, we define $g_k = \nabla f(x_k), s_k = x_{k+1} - x_k, y_k = g_{k+1} - g_k$, H_k is the approximate inverse of $\nabla^2 f(x_k)$. Then, we must have $H_{k+1} y_k = s_k$.

The SR1 method modified H_k by

$$H_{k+1}^{\mathrm{SR1}} = H_k + \frac{(s_k - H_k y_k)(s_k - H_k y_k)^\top}{(s_k - H_k y_k)^\top y_k} \tag{I.2}$$

Algorithm 4 Newton-LM method: minimize f, initial point x_0.

procedure NEWTON_LM(f, x_0)
  $x \leftarrow x_0$
  while not converged **do**
    $G \leftarrow \nabla^2 f(x)$
    $g \leftarrow \nabla f(x)$
    $v \leftarrow 0$
    while $G + vI$ is singular **do**
      if $v = 0$ **then**
        $v \leftarrow \|G\|_2/2$
      else
        $v \leftarrow 2v$
      end if
    end while
    $d = -(G + vI)^{-1} g$
    $\alpha \leftarrow$ LINESEARCH(f, x, d)
    $x \leftarrow x + \alpha d$
  end while
  return x
end procedure

Algorithm 5 Quasi-Newton method: minimize f, initial point x_0, positive definite matrix H

procedure NEWTON_VANILLA(f, x_0)
  $x \leftarrow x_0$
  while not converged **do**
    $G \leftarrow \nabla^2 f(x)$
    $g \leftarrow \nabla f(x)$
    $d \leftarrow -Hg$
    $\alpha \leftarrow$ LINESEARCH(f, x, d)
    $x \leftarrow x + \alpha d$, and modify H.
  end while
  return x
end procedure

The Broyden class formula provides us with the following modification

$$H_{k+1}^{\varphi} = H_{k+1}^{\mathrm{DFP}} + \varphi v_k v_k^{\top} \tag{I.3}$$

where

$$H_{k+1}^{\mathrm{DFP}} = H_k + \frac{s_k s_k^{\top}}{s_k^{\top} y_k} - \frac{H_k s_k s_k^{\top} H_k}{y_k^{\top} H_k y_k} \tag{I.4a}$$

$$v_k = \sqrt{y_k^{\top} H_k y_k} \left(\frac{s_k}{s_k^{\top} y_k} - \frac{H_k y_k}{y_k^{\top} H_k y_k} \right) \tag{I.4b}$$

The selection of φ is subtile. Despite traditionally φ is set to be within $[0, 1]$. We find that φ admits a more flexible choice. We set $\varphi = 0.5, 2$ in our experiments, and do not observe much difference.

Implementation Details The algorithm is unstable in the way that it involves division, where the denominator cannot be properly bound when s_k and y_k are nearly orthogonal or y_k resides in the eigenspace

corresponding to the small eignvalue of H_k. Besides, H_k could be indefinite during the iteration. A more complicated way to deal with this is by Gill amd Murray, we, however, simply set a lower bound for $s_k^\top y_k$ and $y_k^\top H_k y^k$.

II. Line search algorithms

Given x_k and a descent direction d_k, the the line search algorithm aims to find an appropriate step length α_k, such that $f(x_k+\alpha_k d_k)$ has a sufficient decrease compared to $f(x_k)$. We adopt the strong Wolfe criterion

$$f(x_k+\alpha d_k) \le f(x_k)+\rho g_k^\top d_k \alpha \tag{II.1a}$$
$$|g(x_k+\alpha d_k)^\top d_k| \le -\sigma g_k^\top d_k \tag{II.1b}$$

where $1>\sigma>\rho>0$. Equation II.1b can be replace with $g(x_k+\alpha d_k)^\top d_k \ge \sigma g_k^\top d_k$, which leads to Wolfe criterion.

Or, we can adopt Goldstein criterion

$$f(x_k+\alpha d_k) \le f(x_k)+\rho g_k^\top d_k \alpha \tag{II.2a}$$
$$f(x_k+\alpha d_k) \ge f(x_k)+(1-\rho) g_k^\top d_k \alpha \tag{II.2b}$$

where $1>\rho>0$. Equation II.2b can be omitted, which leads to Armijo criterion.

Algorithm 6 Advance-Retreat method

procedure DVANCE_RETREAT($f, x_k, d_k, \alpha_0, \gamma_0, t$)
  $\phi(\alpha) := f(x_k+\alpha d_k)$
  while not converged **do**
    $\alpha_{i+1} \leftarrow \alpha_i + \gamma_i$.
    if $\alpha_{i+1} \le 0$ **then**
      $\alpha_{i+1} \leftarrow 0$, Flag $\leftarrow 1$.
    end if
    if $\phi(\alpha_{i+1}) \ge \phi(\alpha_i)$ **then**
      Flag $\leftarrow 1$.
    end if
    if Flag == 1 **then**
      if $i == 0$ **then**
        $\gamma_i \leftarrow -\gamma_i$, $\alpha \leftarrow \alpha_{i+1}$
      else
        $a \leftarrow \min(\alpha, \alpha_{i+1}), b \leftarrow \max(\alpha, \alpha_{i+1})$
        return a, b
      end if
    else
      $\gamma_{i+1} \leftarrow t\gamma_i, \alpha \leftarrow \alpha_i, i \leftarrow i+1$
    end if
  end while
end procedure

Empirically, we find that we need to rescale direction d_k to get better performace. The scale need to be tuned by experiments.

We use the codes provided by the teacher, which returns the stepsize as well as function evaluation times. We do not modify the codes a lot, and by default, we use inexact search.

The provided codes does not support setting the upper and lower bound for the step size. Empirically, the step size varies a lot during the iteration. Hence, from a greedy view, i.e. to keep the descent for each step as large as possible, we do not set the bound for the step size. Naturally, it will cause some numerical instable error. However, by rescaling d_k properly, we are able to prevent such error. Besides, since the most unstable algorithm is quasi Newton methods, which include division. Hence, we catch this error and set step size to be 1 instead. By using our parameters, such discretion is actually not necessary.

III. Nonlinear least squares problem

We try to minimize $f(x) = \sum_{i=1}^{m} f_i^2(x), x \in \mathbb{R}^n$. The stopping condition of this section is set to be $\max\{\Delta f_k, \|g_k\|\} < 1e-8$, and the number of maximum number of iterations is set to be 1000.

A. Brown and Dennis function

$n = 4, m \geq n$. Define

$$f_i(x) = (x_1 + t_i x_2 - \exp(t_i))^2 + (x_3 + x_4 \sin(t_i) - \cos(t_i))^2 \tag{III.1}$$

where $t_i = i/5$. We start with $x_0 = (25, 5, -5, -1)$, and set $m \in \{4, 10, 20, 30, 40, 50\}$. In this problem, when m is large, the scale of Hessian around the minima point is small, rendering a small scale of d_k. We need to rescale $\|d_k\|_2$ to be 1e-2, and then perform line search.

The results are listed in Table III.1. Clearly, modifed Newton method is not satisfactory. Among Quasi-Newton methods, it seems that BFGS method achieves the best performace.

We find that it is almost impossible to make $\|g_k\| < 10^{-8}$. Hence, we modify our criterion to be $\min\{\Delta f_k, \|g_k\|\} < 1e-8$. Results are presented in Table III.2. Clearly, BFGS outperforms other quasi Newton method, since when $m = 50$, $\|g_k\| = 0.0158$, which is close to the optimal value (see Table III.1). While all Newton type methods (include vanilla Newton), can achieve that accuracy in Table III.2.

As a matter of fact, we find that early stopping for SR1 methods actually benefits performace, since if training to the maximum number of steps will leads $g_k = 8.71e+12$, significantly larger than early stopping results $\|g_k\| = 0.0158$, which indicates that its numerical instability. Hence, optimizing longer does not means better results.

B. Discrete integral equation function

$m = n$. Define

$$\begin{aligned} f_i(x) = x_i + h[(1 - t_i) \sum_{j=1}^{i} t_j (x_j + t_j + 1)^3 \\ + t_i \sum_{j=i+1}^{n} (1 - t_j)(x_j + t_j + 1)^3]/2 \end{aligned} \tag{III.2}$$

where $h = 1/(n+1), t_i = ih$, and $x_0 = x_{n+1} = 0$. We start with $x_0 = (t_j(t_j - 1))$, and the minimum value of f should be 0. We need to rescale $\|d_k\|_2$ to be 1e-2, and then perform line search. Since the problem is well conditioned, almost all method is able to solve it efficiently. However, it seems that Newton's method can achieve lower final value than the quasi Newton method. Perhaps Newton's type method directly uses Hessian while quasi Newton methods are to approximate the Hessian. However, the overall performace is quite close.

TABLE III.1
TEST FUNCTION 1

n		vanilla	damped	mixed	LM	SR1	DFP	BFGS	$\varphi = 0.5$	$\varphi = 2$
	f_k	1.05e-05	1.05e-05	1.05e-05	1.05e-05	1.05e-05	3.61e-05	1.05e-05	1.05e-05	1.05e-05
	$\|g_k\|_2$	4.19e-09	4.76e-09	7.52e-13	4.76e-09	6.18e-09	1.13e-03	4.51e-09	7.24e-09	5.99e-09
4	iter	27	61	133	61	113	1000	112	147	113
	feva	84	320	762	317	531	4952	507	683	512
	time	0.000624	0.01355	0.0216	0.009002	1.74e-02	0.14651	0.02619	0.03166	0.02339
	f_k	1.443225	1.44323	1.44323	1.443225	1.443225	1.44323	1.44323	1.44323	1.44323
	$\|g_k\|_2$	7.39e-09	3.38e-10	3.02e-13	7.33e-09	1.83e-11	5.95e-10	5.95e-09	8.22e-10	7.53e-09
10	iter	25	193	196	64	79	87	75	76	76
	feva	78	1030	1060	332	359	389	320	334	325
	time	0.000618	0.03104	0.0355	0.009104	1.59e-02	0.01577	0.01285	0.01083	0.01028
	f_k	85822.2	85822.2	85822.2	85822.2	85822.2	85822.2	85822.2	85822.2	85822.2
	$\|g_k\|_2$	6.85e-10	2.92e-10	3.11e-11	2.14e-10	8.96e-11	1.05e-09	1.55e-10	6.87e-11	8.31e-11
20	iter	10	13	1000	13	23	24	23	24	23
	feva	33	80	14885	74	120	128	121	115	110
	time	0.000244	0.003	1.0147	0.003532	6.16e-03	0.00787	0.00634	0.00461	0.00531
	f_k	9.77e+08	9.8e+08	9.8e+08	9.77e+08	9.77e+08	9.8e+08	9.8e+08	9.8e+08	9.8e+08
	$\|g_k\|_2$	7.68e-09	7.68e-09	7.68e-09	7.68e-09	7.68e-09	7.68e-09	7.68e-09	7.68e-09	7.68e-09
30	iter	9	66	1000	76	19	26	16	34	18
	feva	30	409	14419	490	116	164	90	211	108
	time	0.000294	0.01823	1.0799	0.029662	1.17e-02	0.01695	0.0067	0.01414	0.01078
	f_k	5.86e+12	5.9e+12	5.9e+12	5.86e+12	5.86e+12	5.9e+12	5.9e+12	5.9e+12	5.9e+12
	$\|g_k\|_2$	4.63e-05	3.86e-05	3.86e-05	4.63e-05	3.86e-05	4.63e-05	4.63e-05	4.63e-05	3.86e-05
40	iter	1000	1000	1000	1000	1000	1000	1000	1000	1000
	feva	3003	14894	14894	14891	12722	13464	13689	13690	13509
	time	0.023504	1.19446	1.18282	1.187792	1.012959	1.10342	1.14791	1.12273	1.0848
	f_k	2.67e+16	2.67e+16	2.67e+16	2.67e+16	2.68e+16	2.67e+16	2.67e+16	2.67e+16	2.67e+16
	$\|g_k\|_2$	0.015815	0.01569	0.01569	0.015815	8.71e+12	0.02703	0.01654	0.01605	0.01569
50	iter	1000	1000	1000	1000	1000	1000	1000	1000	1000
	feva	3003	14932	14932	14621	6458	13142	13084	12972	12976
	time	0.025463	1.23928	1.25334	1.214229	4.11e-01	1.12822	1.12639	1.10689	1.0949

IV. MINIMAL SURFACE PROBLEM

The determination of the surface with minimal area and given boudary values in a convex domain $\mathcal{D}$ is an infinite-dimensional optimization problem of the form

$$\min\{f(v) : v \in K\} \tag{IV.1}$$

where $f : K \to \mathbb{R}$ is the functional

$$f(v) = \int_{\mathcal{D}} \left(1 + \|\nabla v(x)\|^2\right)^{1/2} \mathrm{d}x \tag{IV.2}$$

and the set K is defined by

$$K = \{v \in H^1(\mathcal{D}) : v(x) = v_{\mathcal{D}}(x), \forall x \in \partial\mathcal{D}\}$$

for some boundary data function $v_{\mathcal{D}} : \partial\mathcal{D} \to \mathbb{R}$. The boundary value $v_{\mathcal{D}}$ uniquely define the solution to the minimal surface problem.

We consider Enneper surface by defining $v_{\mathcal{D}}$ on $\mathcal{D} = (-\frac{1}{2}, \frac{1}{2}) \times (-\frac{1}{2}, \frac{1}{2})$ by

$$v_{\mathcal{D}}(\xi_1, \xi_2) = u^2 - v^2 \tag{IV.3}$$

TABLE III.2
TEST FUNCTION 1, LOOSER STOPPING CRITERION

n		vanilla	damped	mixed	LM	SR1	DFP	BFGS	$\varphi = 0.5$	$\varphi = 2$
	f_k	1.05e-05	1.05e-05	1.05e-05	1.05e-05	4.25e-05	5.37e-05	3.63e-05	3.76e-05	3.56e-05
	$\|g_k\|_2$	2.24e-06	2.41e-06	2.96e-08	2.41e-06	1.74e-03	7.29e-04	3.61e-04	3.67e-04	3.59e-04
4	iter	20	57	60	57	86	175	63	62	63
	feva	63	296	314	293	373	829	264	259	264
	time	0.022931	0.02696	0.03095	0.030914	2.38e-02	0.03817	0.01031	0.00874	0.0071
	f_k	1.443225	1.44323	1.44323	1.443225	1.443225	1.44323	1.44323	1.44323	1.44323
	$\|g_k\|_2$	5.01e-05	3.74e-05	5.19e-07	3.74e-05	9.79e-06	6.82e-06	4.59e-07	1.43e-05	9.23e-06
10	iter	18	60	62	60	76	85	74	74	74
	feva	57	311	323	308	341	370	315	315	315
	time	0.001673	0.01061	0.01004	0.011911	1.34e-02	0.01611	0.00891	0.00922	0.00901
	f_k	85822.2	85822.2	85822.2	85822.2	85822.2	85822.2	85822.2	85822.2	85822.2
	$\|g_k\|_2$	2.03e-07	9.70e-08	3.11e-11	1.06e-07	7.52e-08	2.82e-06	1.93e-05	1.41e-07	1.14e-06
20	iter	9	12	15	12	22	22	22	23	22
	feva	30	74	110	68	112	109	107	107	102
	time	0.000675	0.0048	0.00757	0.004209	6.49e-03	0.00491	0.00619	0.00465	0.00399
	f_k	9.77e+08	9.8e+08	9.8e+08	9.77e+08	9.77e+08	9.8e+08	9.8e+08	9.8e+08	9.8e+08
	$\|g_k\|_2$	1.50e-06	7.68e-09	7.68e-09	7.68e-09	7.68e-09	8.30e-09	7.68e-09	7.68e-09	7.68e-09
30	iter	7	65	66	75	19	25	16	33	17
	feva	24	394	409	475	116	150	90	197	94
	time	0.000817	0.01699	0.01762	0.020985	6.74e-03	0.01771	0.00565	0.01227	0.00651
	f_k	5.86e+12	5.9e+12	5.9e+12	5.86e+12	5.86e+12	5.9e+12	5.9e+12	5.9e+12	5.9e+12
	$\|g_k\|_2$	4.63e-05	3.86e-05	3.86e-05	4.63e-05	4.63e-05	4.63e-05	4.63e-05	4.63e-05	1.18e-01
40	iter	10	15	15	16	32	60	35	35	53
	feva	33	119	119	131	193	304	179	180	251
	time	0.005292	0.01063	0.01131	0.013367	0.025302	0.02055	0.01161	0.0087	0.01464
	f_k	2.67e+16	2.67e+16	2.67e+16	2.67e+16	2.67e+16	2.67e+16	2.67e+16	2.67e+16	2.67e+16
	$\|g_k\|_2$	0.015815	0.01569	0.01569	0.015815	0.015815	125.73	0.0157	568536	0.98474
50	iter	13	9	9	52	67	42	111	97	25
	feva	42	67	67	401	387	222	638	541	143
	time	0.001007	0.00822	0.00738	0.039044	2.77e-02	0.01293	0.03919	0.03883	0.01029

where u, v are the unique solution to the equation

$$\xi_1 = u^2 + uv - \frac{1}{3}u^3, \quad \xi_2 = -v - u^2 v + \frac{1}{3}v^3 \tag{IV.4}$$

A finite element approximation to the minimal surface problem is obtained by minimize f over the space of piecewise linear function v with values $v_{i,j}$ at $z_{i,j}$, where $z_[i,j] \in \mathbb{R}^2$ are the vertice of a triangulation of $\mathcal{D}$ with grid spacings h_x and h_y. The value $v_{i,j}$ is obtained by solving the minimization problem

$$\min\{\sum \left(f_{i,j}^L(v) + f_{i,j}^U(v)\right) : v \in \mathbb{R}\} \tag{IV.5}$$

where

$$f_{i,j}^L(v) = \frac{h_x h_y}{2}\left[1 + \left(\frac{v_{i+1,j} - v_{i,j}}{h_x}\right)^2 + \left(\frac{v_{i,j+1} - v_{i,j}}{h_x}\right)^2\right]^{1/2}$$

$$f_{i,j}^U(v) = \frac{h_x h_y}{2}\left[1 + \left(\frac{v_{i-1,j} - v_{i,j}}{h_x}\right)^2 + \left(\frac{v_{i,j-1} - v_{i,j}}{h_x}\right)^2\right]^{1/2}$$

TABLE III.3
TEST FUNCTION 2

n		vanilla	damped	mixed	LM	SR1	DFP	BFGS	$\varphi = 0.5$	$\varphi = 2$
2	f_k	6.95e-22	1.11e-20	3.62e-27	1.11e-20	1.86e-14	1.20e-17	1.06e-17	1.13e-17	9.72e-18
	$\|g_k\|_2$	6.76e-11	2.73e-10	1.56e-13	2.73e-10	3.59e-07	9.14e-09	8.61e-09	8.85e-09	8.23e-09
	iter	3	6	8	6	5	6	6	6	6
	feva	12	38	50	35	26	31	31	31	31
	time	0.000294	0.00304	0.01183	0.00423	1.61e-03	0.00195	0.00163	0.00181	0.0016
10	f_k	3.99e-22	2.60e-21	2.83e-29	2.60e-21	1.18e-12	6.82e-16	1.14e-15	8.84e-16	1.75e-15
	$\|g_k\|_2$	4.95e-11	1.28e-10	1.33e-14	1.28e-10	2.66e-06	6.03e-08	7.82e-08	6.89e-08	9.60e-08
	iter	3	6	8	6	6	8	8	8	8
	feva	12	37	49	34	30	40	40	40	40
	time	0.00023	0.00253	0.00584	0.00201	2.03e-03	0.00269	0.00216	0.00353	0.0028
20	f_k	6.59e-22	3.87e-21	1.46e-29	3.87e-21	1.32e-12	7.59e-16	2.54e-15	1.43e-15	5.81e-15
	$\|g_k\|_2$	6.34e-11	1.56e-10	9.67e-15	1.56e-10	2.84e-06	6.72e-08	1.22e-07	9.19e-08	1.82e-07
	iter	3	7	9	7	7	9	9	9	9
	feva	12	42	54	39	34	44	44	44	44
	time	0.000291	0.00302	0.00456	0.00438	2.38e-03	0.00478	0.00422	0.00413	0.00312
30	f_k	9.43e-22	1.50e-17	1.19e-26	1.50e-17	2.58e-12	3.57e-17	6.85e-17	4.47e-17	1.38e-16
	$\|g_k\|_2$	7.59e-11	9.60e-09	2.70e-13	9.60e-09	3.71e-06	1.51e-08	2.08e-08	1.69e-08	2.98e-08
	iter	3	6	8	6	7	10	10	10	10
	feva	12	36	48	33	34	48	48	48	48
	time	0.000477	0.00289	0.00774	0.0033	2.60e-03	0.00395	0.00794	0.00473	0.00442
40	f_k	1.23e-21	3.29e-18	3.42e-27	3.29e-18	1.63e-11	6.72e-18	2.94e-17	1.48e-17	5.60e-17
	$\|g_k\|_2$	8.68e-11	4.57e-09	1.47e-13	4.57e-09	8.24e-06	5.83e-09	1.23e-08	8.78e-09	1.66e-08
	iter	3	7	9	7	7	10	10	10	10
	feva	12	41	53	38	33	48	48	48	48
	time	0.000759	0.00711	0.01268	0.00608	0.00612	0.01059	0.00963	0.01155	0.01368
50	f_k	1.53e-21	4.24e-18	1.51e-27	4.24e-18	1.99e-11	5.11e-17	1.46e-16	1.19e-16	1.94e-16
	$\|g_k\|_2$	9.66e-11	5.13e-09	9.68e-14	5.13e-09	9.49e-06	1.72e-08	3.08e-08	2.78e-08	3.56e-08
	iter	3	7	9	7	7	10	11	11	11
	feva	12	41	53	38	33	48	52	52	52
	time	0.001633	0.01291	0.01562	0.00905	7.78e-03	0.01397	0.01351	0.01472	0.01534

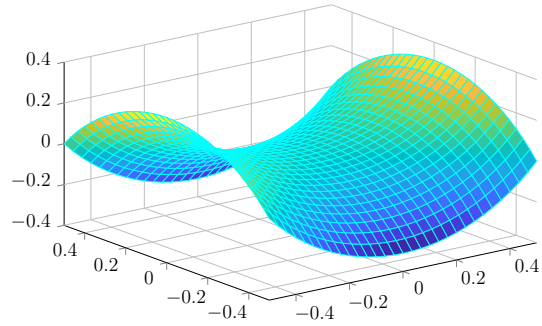
Note that $f^L_{i,j}$ is defined when $0 \le i \le n_x$ and $0 \le j \le n_y$, while $f^U_{i,j}$ is defined when $1 \le i \le n_x + 1$ and $1 \le j \le n_y + 1$.

Note that IV.4 does not necessarily have unique solutions (usually up to 5 real solutions). If the solver converges to a relatively far position, the surface will appear to have spikes on its boundaries. We choose $(u_0, v_0) = (0, 0)$ as initial value and use built-in method "fsolve" to obtain the boundary value. Besides, the iteration start with $v_0 = 0$, i.e. all values attached to inner grid point is set to zero in the beginning. If $n = 5$, we resize $\|d_k\|_2$ to be $1e-4$ to obtain better performace. Otherwise, we resize $\|d_k\|_2 = 1e-6$.

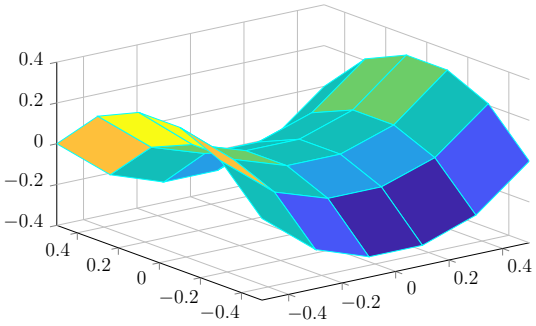
We also modify the stopping criterion to be $|f_k - f_{k-1}| < 1e-8$ or $\|g_k\| < 1e-8$.

We plot the results obtained by BFGS method in IV.1, which utilize "Enneper_surface.m". All results are presented in Table IV.1. We find that generally, Broyden class method outperforms SR1 method, which is not counterintuitive since Broyden class use second order rank modification. Within the Broyden class method, the difference is not significant.

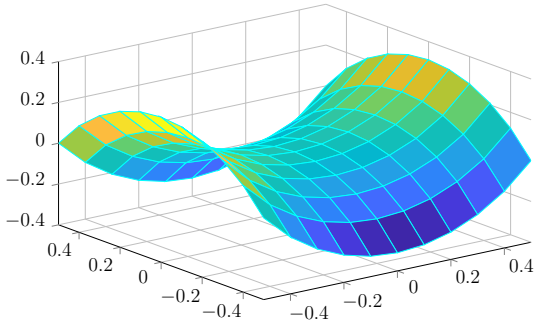
An interesting observation is that obtaining Enneper surface by optimization or direct computation cost similar budgets. Since we are required to solve a nonlinear equation on each grid point in the latter case.



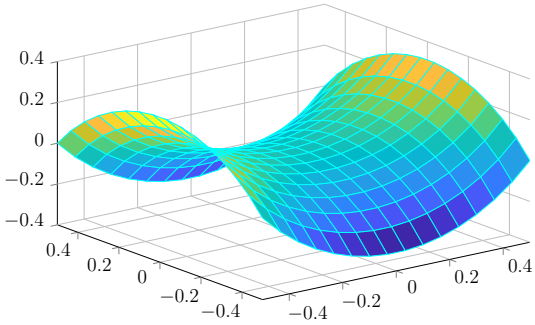
(a) Groundtruth



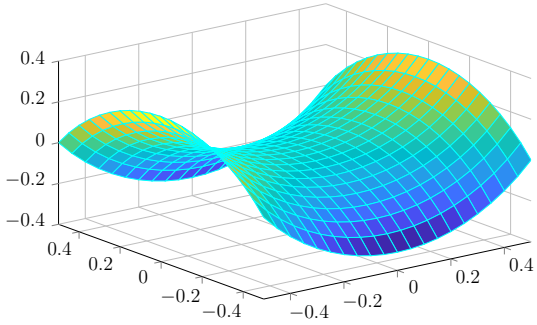
(b) BFGS, $n_x = n_y = 5$



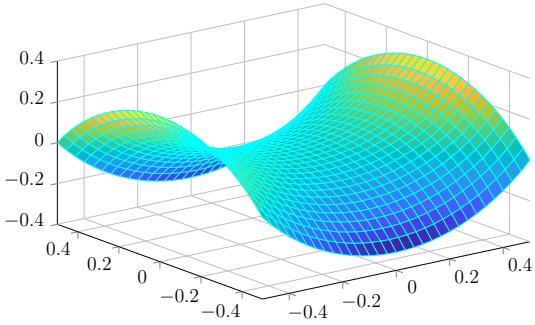
(c) BFGS, $n_x = n_y = 10$



(d) BFGS, $n_x = n_y = 15$



(e) BFGS, $n_x = n_y = 20$



(f) BFGS, $n_x = n_y = 30$

Fig. IV.1. Enneper surface, BFGS method

TABLE IV.1
MINIMAL SURRFACE, QUASI-NEWTON METHOD

n		SR1	DFP	BFGS	$\varphi = 0.5$	$\varphi = 2$
	f_k	1.40731	1.40731	1.40731	1.40731	1.40731
	$\|g_k\|_2$	7.42e-09	5.15e-09	6.85e-09	6.15e-09	7.77e-09
5	iter	5	5	5	5	5
	feva	26	26	26	26	26
	time	0.037291	0.00509	0.00251	0.002749	0.004647
	f_k	1.417862	1.41786	1.41786	1.417862	1.417862
	$\|g_k\|_2$	0.000187	0.00042	0.00024	0.000348	0.00013
10	iter	14	11	11	11	11
	feva	89	56	56	56	56
	time	0.010513	0.01054	0.00799	0.006574	0.008763
	f_k	1.419807	1.41981	1.41981	1.419807	1.419807
	$\|g_k\|_2$	0.000167	0.00017	0.00036	0.000386	0.000184
15	iter	18	16	15	15	16
	feva	118	81	76	76	81
	time	0.025988	0.0116	0.01055	0.014857	0.012345
	f_k	1.420488	1.42049	1.42049	1.420488	1.420488
	$\|g_k\|_2$	0.000238	0.0002	0.00026	0.000276	0.00035
20	iter	22	21	20	20	19
	feva	138	106	101	101	96
	time	0.05226	0.03921	0.03384	0.036325	0.030037
	f_k	1.420974	1.42097	1.42097	1.420974	1.420974
	$\|g_k\|_2$	0.000644	0.00025	0.00021	0.000221	0.000282
30	iter	31	33	32	32	31
	feva	238	166	161	161	156
	time	0.208058	0.34491	0.36279	0.342336	0.338139
	f_k	1.421144	1.42114	1.42114	1.421144	1.421144
	$\|g_k\|_2$	0.000917	0.00017	0.00016	0.000156	0.000188
40	iter	40	45	44	44	43
	feva	301	226	221	221	216
	time	0.771245	1.51432	1.46009	1.472486	1.438696