Solving Waypoint-Constrained Optimal Control Problem via Interior-point Method

Dong Ho Lee

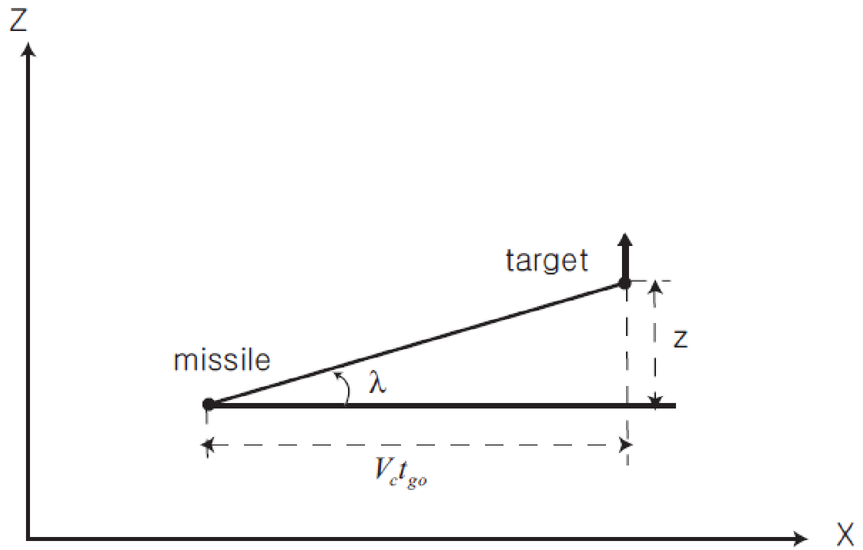
KAIST Aerospace Engineering

발표 일시: 2021년 6월 15일 21:00

Contents

Contents

Homing Guidance Problem



Equations of Motion (EOM):

$$\boldsymbol{x} = \begin{bmatrix} z \\ v \end{bmatrix} \in R^n$$

$$\dot{\boldsymbol{x}} = \begin{bmatrix} v \\ -u+g \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} 0 \\ -u+g \end{bmatrix}$$

- $z(t), v(t)$: Relative position and velocity of target w.r.t missile along Z-axis.
- Initial target range $R_0 = 3000m$
- Missile horizontal velocity $V_c = 300ms^{-1}$
- $z(t_f) = v(t_f) = 0$ for a successful intercept.
- Objective: Minimize the overall control effort throughout the trajectory.

From Optimal Control problem to Parameter Optimization

trajectory optimization problem → constrained parameter optimization

- decision variables are vector ***functions***
- infinite dimensional
- ***differential*** equations

- decision variables are real ***numbers***
- finite dimensional
- ***algebraic*** equations

Indirect Methods ⟷ **Direct Methods**

- "optimize then discretize"
- more accurate
- harder to pose and solve

- "discretize the optimize"
- less accurate
- easier to pose and solve

Shooting Methods ⟷ **Collocation Methods**

- based on simulation
- better for problems with simple control and no path constraints

- based on function approximation
- better for problems with complicated control and/or path constraints

https://www.youtube.com/watch?v=wlkRYMVUZTs

Three Direct Collocation Methods

- Euler Method
- Trapezoidal Method

Treated in Interim Report

- Hermite-Simpson Method
  - Medium-order direct collocation
  - State represented by cubic Hermite splines
  - Control is assumed to be piecewise-linear
  - Dynamics satisfied using Simpson quadrature
  - Midpoint control and state values required
  - Approximates the dynamics $\dot{\boldsymbol{x}} = \boldsymbol{f}(t, \boldsymbol{x}, \boldsymbol{u})$ as $x_{k+1} = x_k + \frac{h}{6}\left[f_k + 4f_c + f_{k+1}\right]$

where $x_c = \frac{1}{2}(x_k + x_{k+1}) + \frac{h}{8}(f_k - f_{k+1})$, $u_c = \frac{1}{2}(u_k + u_{k+1})$

Optimization Problem using Hermite-Simpson Collocation

Objective	$\min J = \frac{t_f}{N}\sum_{i=0}^{N-1}\left[\frac{1}{3}u_i^2 + \frac{1}{3}u_i u_{i+1} + \frac{1}{3}u_{i+1}^2\right]$
Decision Variables	$z_0, v_0, \ldots, z_N, v_N, u_0, \ldots, u_N$
Dynamic Constraints	$z_{k+1} = z_k + \frac{h}{6}(v_k + 4v_c + v_{k+1})$ $v_c = \frac{1}{2}(v_k + v_{k+1}) + \frac{h}{8}(-u_k + u_{k+1})$ $v_{k+1} = v_k + \frac{h}{6}(-u_k - 4u_c - u_{k+1} + 6g)$ $u_c = \frac{1}{2}(u_k + u_{k+1})$
Initial and Final Conditions	$\boldsymbol{x}_0 = \begin{bmatrix}100\\10\end{bmatrix}, \boldsymbol{x}_N = \begin{bmatrix}0\\0\end{bmatrix}$

Minimize	$\min J = \frac{t_f}{N}\sum_{i=0}^{N-1}\left[\frac{1}{3}u_i^2 + \frac{1}{3}u_i u_{i+1} + \frac{1}{3}u_{i+1}^2\right]$
Equality Constraints	$Ax = b, \quad A \in R^{(2N+4)\times(3N+3)}, \; b \in R^{(2N+4)\times 1}$
Inequality Constraints	$Cx \le d,$ $C = \begin{bmatrix} I_{(3N+3)} \\ -I_{(3N+3)} \end{bmatrix} \in R^{(6N+6)\times(3N+3)}, \; \mathrm{d} \in R^{(6N+6)\times 1}$

Upper bounds	$z_k, v_k, u_k \le 1000$
Lower bounds	$-z_k, -v_k, -u_k \le 1000$
Number of Equality Constraints	$2N+4$
Number of Inequality Constraints	$6N+6$

Addition of Waypoint Constraints

Objective	$\min J = \frac{t_f}{N}\sum_{i=0}^{N-1}\left[\frac{1}{3}u_i^2 + \frac{1}{3}u_i u_{i+1} + \frac{1}{3}u_{i+1}^2\right]$
Decision Variables	$z_0, v_0, \ldots, z_N, v_N, u_0, \ldots, u_N$
Dynamic Constraints	$z_{k+1} = z_k + \frac{h}{6}(v_k + 4v_c + v_{k+1})$ $v_c = \frac{1}{2}(v_k + v_{k+1}) + \frac{h}{8}(-u_k + u_{k+1})$ $v_{k+1} = v_k + \frac{h}{6}(-u_k - 4u_c - u_{k+1} + 6g)$ $u_c = \frac{1}{2}(u_k + u_{k+1})$
Initial and Final Conditions	$\boldsymbol{x}_0 = \begin{bmatrix}100\\10\end{bmatrix}, \boldsymbol{x}_N = \begin{bmatrix}0\\0\end{bmatrix}$
Waypoint Constraints	$z(t_1) = z_1,\ z(t_2) = z_2$ $0 \le t_1, t_2 \le 10$

Minimize	$\min J = \frac{t_f}{N}\sum_{i=0}^{N-1}\left[\frac{1}{3}u_i^2 + \frac{1}{3}u_i u_{i+1} + \frac{1}{3}u_{i+1}^2\right]$
Equality Constraints	$Ax = b,\quad A \in R^{(2N+6)\times(3N+3)},\ b \in R^{(2N+6)\times 1}$
Inequality Constraints	$Cx \le d,$ $C = \begin{bmatrix} I_{(3N+3)} \\ -I_{(3N+3)} \end{bmatrix} \in R^{(6N+6)\times(3N+3)},\ \mathrm{d} \in R^{(6N+6)\times 1}$

Upper bounds	$z_k, v_k, u_k \le 1000$
Lower bounds	$-z_k, -v_k, -u_k \le 1000$
Number of Equality Constraints	$2N+6$
Number of Inequality Constraints	$6N+6$

Contents

$$\min_x \; f(x)$$

$$x \in R^n, f: R^n \to R$$

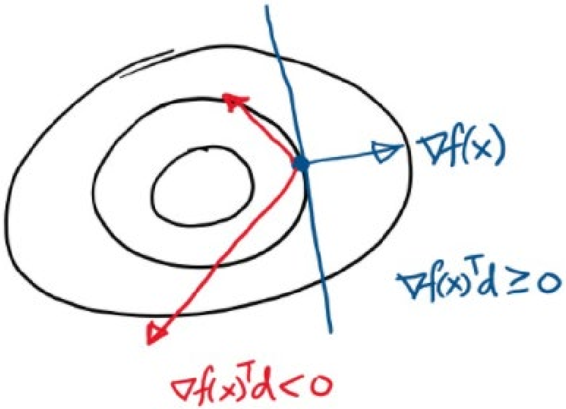
Steepest Descent

- The simplest method for unconstrained optimization is steepest descent.
- **Key idea:** The negative gradient $-\nabla f(x)$ points in the "steepest downhill" direction for $f(x)$ at x.
- **Question:** How far should we go in the direction of $-\nabla f(x_k)$?
- **Line Search:** For a direction $s \in R^n$, let $\phi: R \to R$ be $\phi(\eta) = f(x + \eta s)$. Then, minimizing f along s corresponds to minimizing the one-dimensional function $\phi(\eta)$.
  - Golden Section Line Search
  - Backtracking Line Search

STEEPEST DESCENT

Choose initial guess x_0, *convergence tolerance* tol
for $k = 0, 1, 2, \dots$ **do**
  $s_k = -\nabla f(x_k)$
  if $\|\nabla f(x_k)\|_2 \le tol$ **then** converged
  Choose η_k that minimizes $\phi(\eta_k) = f(x_k + \eta_k s_k)$
  $x_{k+1} \leftarrow x_k + \eta_k s_k$
end for

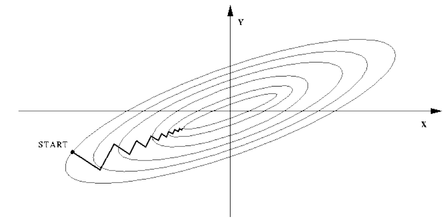
$$f(x+d) \approx f(x) + \nabla f(x)^T d$$



Newton's Method

- Steepest descent often converges **very slowly**.
  - Linear convergence rate, zigzag pattern
  - Not suitable for badly scaled problem where the eigenvalues of the Hessian

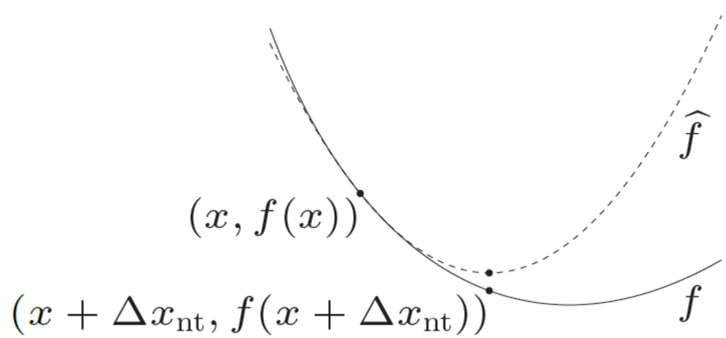
  at the solution are different by several orders of magnitude; large $\kappa = {}^{\lambda_{max}}/_{\lambda_{min}}$



- **Key idea:** We can get faster convergence by using more information about $f(x)$
- **Motivation:** $f(x_k + s_k) \approx f(x_k) + \nabla f(x_k)^T s_k + \frac{1}{2} s_k \nabla^2 f(x_k) s_k$
- s_k should minimize $\hat{f}(x_k + s_k) = f(x_k) + \nabla f(x_k)^T s_k + \frac{1}{2} s_k \nabla^2 f(x_k) s_k$
- So, $\nabla_{s_k} \hat{f}(x_k + s_k) = \nabla f(x_k) + H_k s_k = 0 \Rightarrow \underline{s_k = -H_k^{-1} \nabla f(x_k)}$, $\; H_k = \nabla^2 f(x_k)$

NEWTON'S METHOD

```
Choose initial guess x_0, convergence tolerance tol
for k = 0, 1, 2, ... do
    Solve H_k s_k = -∇f(x_k) for s_k
    if ||∇f(x_k)||_2 ≤ tol then converged
    Choose η_k that minimizes φ(η_k) = f(x_k + η_k s_k)
    x_{k+1} ← x_k + η_k s_k
end for
```



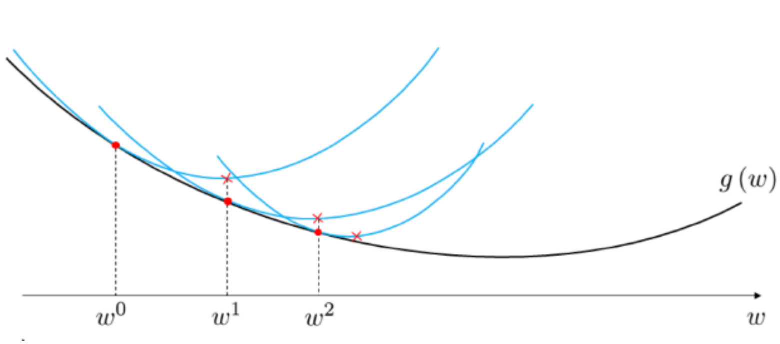
$\boxed{s_k = \Delta x_{nt} \quad \text{(Newton step)}}$

Newton's Method

- Searches for the **stationary points** of quadratic approximation of the function,

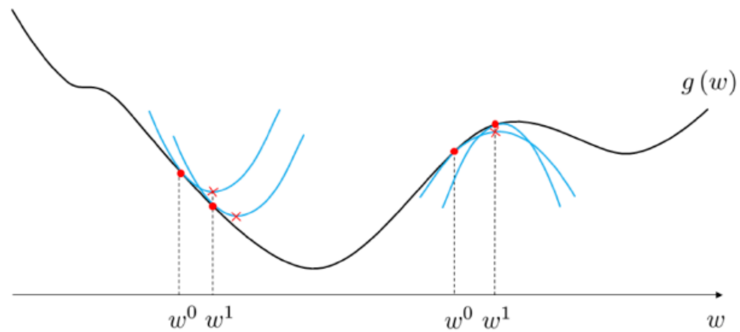
$$\hat{f}(x_k + s_k) = f(x_k) + \nabla f(x_k)^T s_k + \frac{1}{2} s_k \nabla^2 f(x_k) s_k$$

- Convex Problem



For convex functions, these approximations are **always convex** and so their stationary points are minima.

- Non-Convex Problem



For non-convex functions, quadratic approximations can be **concave** or **convex** depending on where they are constructed, leading the algorithm to possibly converge to a maximum.

- Problems with Newton's method:
  - Only converges when sufficiently close to a minimum, the Hessian is dense in general and so very expensive to compute its inverse if n is large, can be impractical to derive the Hessian analytically

Quasi-Newton's Method

- Quasi-Newton's method **do not** require the Hessian matrix.
- Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm
  - Computes the approximation of Hessian iteratively
  - Improves the Hessian approximation using gradient evaluations
- Search direction: $B_k s_k = -\nabla f(x_k),\ B_k \approx H_k$
- Quasi-Newton (secant) condition:
  - Let $\tilde{s}_k \equiv x_{k+1} - x_k = \eta_k s_k,\ y_k \equiv \nabla f(x_{k+1}) - \nabla f(x_k)$
  - Note: $\nabla f(x_{k+1}) = \nabla f(x_k + \tilde{s}_k) = \nabla f(x_k) + \nabla^2 f(x_k)\tilde{s}_k$
  - Then, $\nabla^2 f(x_k)\tilde{s}_k \approx \nabla f(x_{k+1}) - \nabla f(x_k) = y_k$
  - Thus, B_{k+1} must satisfy $B_{k+1}\tilde{s}_k = y_k$
- Symmetric Rank-Two Update Formula (from Lecture note)
  - $B_{k+1} = B_k - \frac{1}{\tilde{s}_k^T B_k \tilde{s}_k} B_k \tilde{s}_k \tilde{s}_k B_k^T + \frac{1}{y_k^T \tilde{s}_k} y_k y_k^T$
- Inverse Hessian Approach (from Lecture note)
  - Motivation: $s_k = -B_k^{-1}\nabla f(x_k) = -\widetilde{H}_k \nabla f(x_k)$
  - $\widetilde{H}_{k+1} = \widetilde{H}_k + \frac{1}{y_k^T \tilde{s}_k}\tilde{s}_k\,\tilde{s}_k^T - \frac{1}{y_k^T \widetilde{H}_k y_k}\widetilde{H}_k y_k \left(\widetilde{H}_k y_k\right)^T + \left(y_k \widetilde{H}_k y_k\right) v_k v_k^T$ where $v_k = \frac{\tilde{s}_k}{y_k^T y_k} - \frac{H_k y_k}{y_k^T \widetilde{H}_k y_k}$

Quasi-Newton's Method

- Inverse Hessian Approach (from Lecture note)
  - Motivation: $s_k = -B_k^{-1}\nabla f(x_k) = -\widetilde{H}_k \nabla f(x_k)$
  - $\widetilde{H}_{k+1} = \widetilde{H}_k + \frac{1}{y_k^T \tilde{s}_k}\tilde{s}_k\,\tilde{s}_k^T - \frac{1}{y_k^T \widetilde{H}_k y_k}\widetilde{H}_k y_k\left(\widetilde{H}_k y_k\right)^T + \left(y_k \widetilde{H}_k y_k\right) v_k v_k^T$ where $v_k = \frac{\tilde{s}_k}{y_k^T y_k} - \frac{H_k y_k}{y_k^T \widetilde{H}_k y_k}$

BFGS ALGORITHM

Choose initial guess x_0, *convergence tolerance* tol, $\widetilde{H}_0 = I$
while $\left\|\nabla f(x_k)\right\|_2 \geq tol$ ***do***
  $s_k = -\widetilde{H}_k \nabla f(x_k)$
  Choose η_k *that minimizes* $\phi(\eta_k) = f(x_k + \eta_k s_k)$ *by BTLS*
  $x_{k+1} \leftarrow x_k + \eta_k s_k$ *or* $x_{k+1} \leftarrow x_k - \eta_k \widetilde{H}_k \nabla f(x_k)$
  $\tilde{s}_k = x_{k+1} - x_k$
  $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$
  $\rho_k = \frac{1}{y_k^{\,T} s_k}$
  $\widetilde{H}_{k+1} \leftarrow \left(I - \rho_k \tilde{s}_k y_k^{\,T}\right)\widetilde{H}_k\left(I - \rho_k y_k \tilde{s}_k^{\,T}\right) + \rho_k \tilde{s}_k \tilde{s}_k^{\,T}$
end while

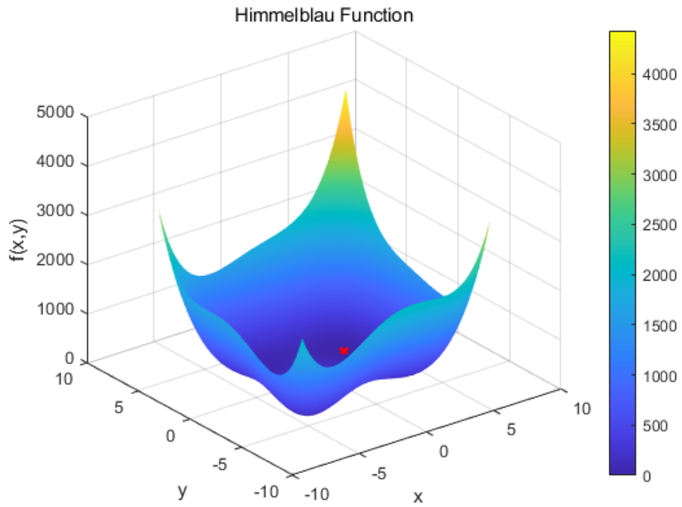
- Backtracking Line Search
  1. "Sufficient decrease condition"
     - $f(x_{k+1}) \leq f(x_k) + c_1 \eta \nabla f(x_k)^T s_k$
     - $c_1 \epsilon\ (0,1)$
  2. "Curvature condition"
     - $\nabla f(x_{k+1})^T s_k \geq c_2 \eta \nabla f(x_k)^T s_k$
     - $c_2 \epsilon\ (c_1, 1)$

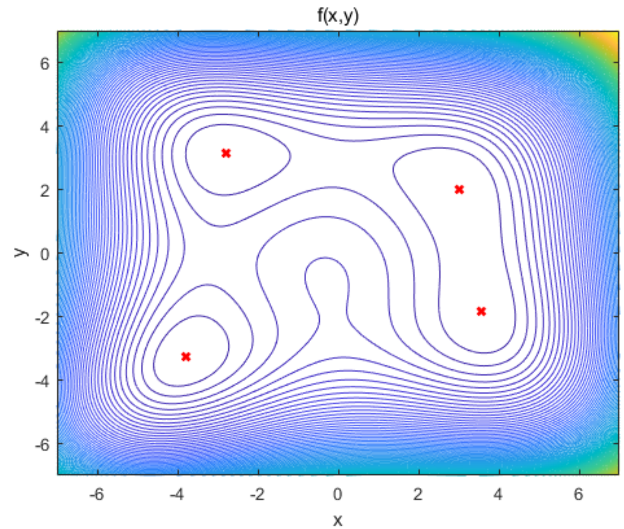
Interim Summary

- Unconstrained Optimization Problem
- Steepest Gradient Descent: $x_{k+1} = x_k - \eta \, I \, \nabla f(x_k)$
- Newton's Method : $x_{k+1} = x_k - \eta \, [\nabla^2 f(x_k)\ ]^{-1} \, \nabla f(x_k)$
- Quasi-Newton's method: $x_{k+1} = x_k - \eta \, \widetilde{H}_k \, \nabla f(x_k)$

Test on Benchmark Function

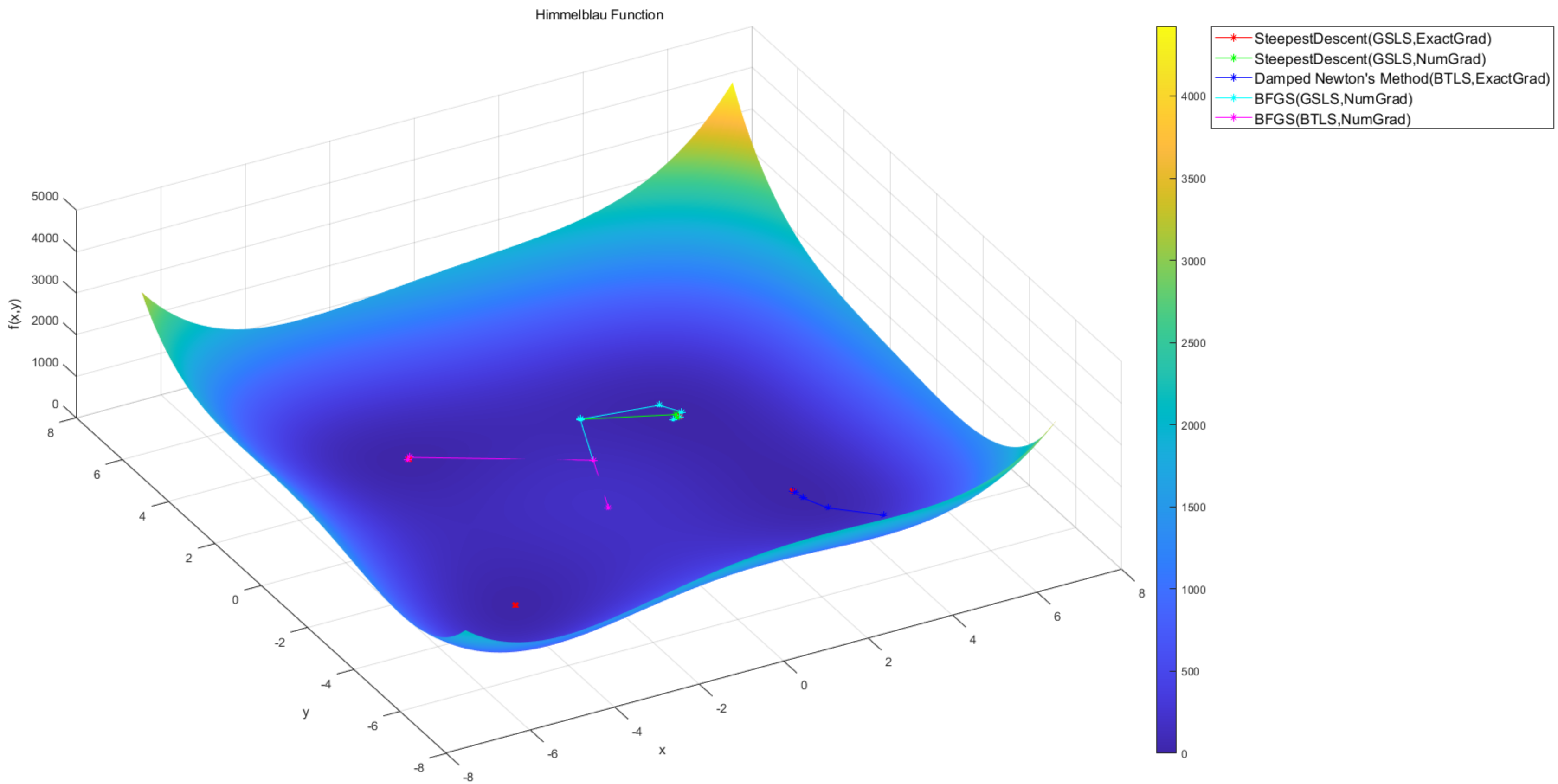
- Himmelblau Function $f(x, y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2$
- Continuous, non-convex, multimodal function
- Used to test optimization algorithms
- It has one local maximum at $x = -0.270845$ and $y = -0.923039$ where $f(x, y)$ =181.617
- It has four identical local minima:
  - $f(3, 2) = f(-2.805188, 3.131312) = f(-3.779310, -3.283186) = f(3.584428, -1.848126) = 0$





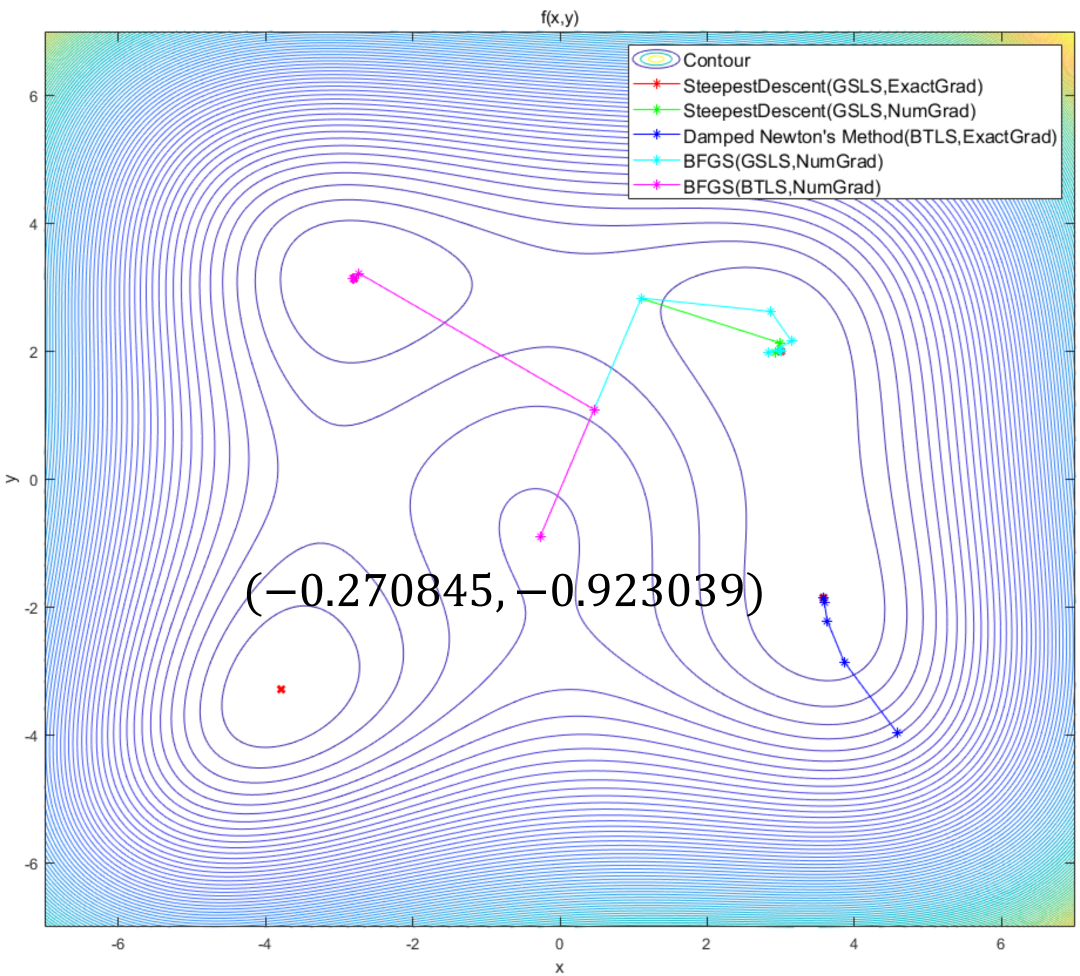
Test on Benchmark Function

- Implemented Steepest Descent, (Damped) Newton's Method, and BFGS algorithm



Test on Benchmark Function

- Implemented Steepest Descent, (Damped) Newton's Method, and BFGS algorithm



Contents

$$\min_x \; f(x) \;\; s.t. \;\; h(x)=0$$

$$x \in R^n, \; f: R^n \to R, \; h: R^p \to R$$

Constrained Newton's Method

- Equality Constrained Problem
  - $\min_{x \epsilon R^n} f(x)$ subject to $h(x) = 0$
  - Lagrangian: $L(x, \lambda) = f(x) + \lambda^T h(x)$
  - Lagrange multiplier: $\lambda \epsilon R^p$
- Optimality Conditions
  - At optimal point, $\nabla L(x^*, \lambda^*) = 0$
  - $\nabla L(x^*, \lambda^*) = \begin{bmatrix} \nabla_x L(x^*, \lambda^*) \\ \nabla_\lambda L(x^*, \lambda^*) \end{bmatrix} = \begin{bmatrix} \nabla_x f(x^*) + \nabla_x h(x^*)\lambda^* \\ h(x^*) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ ← n equations, ← p equations
- Two ways to derive the Newton step Δx_{nt}
  1. Solution to the approximate quadratic problem
  2. Solution to the linearized optimality conditions

1. Newton step via Second-order Approximation

- The Newton step Δx_{nt} solves the linearized (convex quadratic) problem
  - $\min_{\Delta x \epsilon R^n} \ \hat{f}(x+\Delta x) = f(x) + \nabla f(x)^T \Delta x + \frac{1}{2}\Delta x^T \nabla^2 f(x)\, \Delta x$
    s.t. $\hat{h}(x+\Delta x) = h(\cancel{x}) + Dh(x)\,\Delta x = 0$

$$\boxed{Dh(x) = \begin{bmatrix} Dh_1(x) \\ \vdots \\ Dh_p(x) \end{bmatrix} = \begin{bmatrix} \nabla h_1(x)^T \\ \vdots \\ \nabla h_p(x)^T \end{bmatrix} = \nabla h(x)^T \quad (p \times n)}$$

- The Lagrangian for this problem is
  - $L(\Delta x, \lambda) = f(x) + \nabla f(x)^T \Delta x + \frac{1}{2}\Delta x^T \nabla^2 f(x)\,\Delta x + \lambda^T (Dh(x)\,\Delta x)$

- Optimality Conditions: $\nabla L(x, \lambda) = 0$
  - $\nabla L(\Delta x, \lambda) = \begin{bmatrix} \nabla_{\Delta x} L(\Delta x, \lambda) \\ \nabla_\lambda L(\Delta x, \lambda) \end{bmatrix} = \begin{bmatrix} \nabla f(x) + \nabla^2 f(x)\,\Delta x + Dh(x)^T \lambda \\ Dh(x)\,\Delta x \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$
  - Equivalently, $\begin{bmatrix} \nabla^2 f(x) & Dh(x)^T \\ Dh(x) & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda \end{bmatrix} = \begin{bmatrix} -\nabla f(x) \\ 0 \end{bmatrix}$ … "KKT System"

$$\boxed{\nabla_x(Ax) = [D(Ax)]^T = A^T}$$

  - Suppose, we have **linear equality constraints**, i.e. $h(x) = Ax - b = \begin{bmatrix} a_1^T x - b_1 \\ \vdots \\ a_p^T x - b_p \end{bmatrix} = 0$. Then,

$$Dh(x) = \begin{bmatrix} D(a_1^T x - b_1) \\ \vdots \\ D(a_p^T x - b_p) \end{bmatrix} = \begin{bmatrix} a_1^T \\ \vdots \\ a_1^T \end{bmatrix} = A \Rightarrow \boxed{\underbrace{\begin{bmatrix} \nabla^2 f(x) & A^T \\ A & 0 \end{bmatrix}}_{S} \begin{bmatrix} \Delta x \\ \lambda \end{bmatrix} = \begin{bmatrix} -\nabla f(x) \\ 0 \end{bmatrix}}$$

$$\boxed{D(a_i^T x - b_i) = \begin{bmatrix} \frac{\partial (a_i^T x)}{\partial x_1} & \cdots & \frac{\partial (a_i^T x)}{\partial x_n} \end{bmatrix} = a_i^T}$$

2. Newton step via Linearized Optimality Conditions

- Optimality conditions (replace $x^* \leftarrow x + \Delta x_{nt}$)
  - $\nabla L(x^*,\lambda^*) = \begin{bmatrix} \nabla_x L(x^*,\lambda^*) \\ \nabla_\lambda L(x^*,\lambda^*) \end{bmatrix} = \begin{bmatrix} \nabla_x f(x^*) + \nabla_x h(x^*)\lambda^* \\ h(x^*) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ becomes

$$\begin{bmatrix} \nabla_x f(x+\Delta x_{nt}) + \nabla_x h(x+\Delta x_{nt})\lambda \\ h(x+\Delta x_{nt}) \end{bmatrix} = \begin{bmatrix} \nabla_x f(x) + \nabla_x^2 f(x)\Delta x_{nt} + \nabla_x h(x)\lambda + \cancel{\nabla_x^2 h}(x)\Delta x_{nt}\lambda \\ \cancel{h(x)} + \nabla_x h(x)^T\lambda \end{bmatrix}$$

$$= \begin{bmatrix} \nabla_x f(x) + \nabla_x^2 f(x)\Delta x_{nt} + Dh(x)^T\lambda \\ Dh(x)\lambda \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \text{ since } \nabla_x h(x) = Dh(x)^T$$

  - Thus, we have

$$\boxed{\begin{bmatrix} \nabla^2 f(x) & Dh(x)^T \\ Dh(x) & 0 \end{bmatrix} \begin{bmatrix} \Delta x_{nt} \\ \lambda \end{bmatrix} = \begin{bmatrix} -\nabla f(x) \\ 0 \end{bmatrix}}$$

… "KKT System"

$$\nabla_x h(x+\Delta x_{nt}) = Dh(x+\Delta x_{nt})^T = \begin{bmatrix} Dh_1(x+\Delta x_{nt}) \\ \vdots \\ Dh_p(x+\Delta x_{nt}) \end{bmatrix}^T = \begin{bmatrix} Dh_1(x) + D\nabla h_1(x)^T\Delta x_{nt} \\ \vdots \\ Dh_p(x) + D\nabla h_p(x)^T\Delta x_{nt} \end{bmatrix}^T$$

$$= \begin{bmatrix} Dh_1(x\ ) \\ \vdots \\ Dh_p(x) \end{bmatrix}^T + D\nabla h(x)^T\Delta x_{nt} = \nabla h(x) + \nabla^2 h(x)\Delta x_{nt}$$

Infeasible start Newton's Method

- The previous interpretation can be extended to Newton step at infeasible points.
- Assume linear equality constraints, i.e. $h(x) = Ax - b = 0$.
- Let x' denote the current point, **not necessarily feasible**, i.e. $Ax' - b \neq 0,\ x' \in dom\ f$.
- Optimality conditions (replace $x^* \leftarrow x' + \Delta x_{nt}$)
  - $\begin{bmatrix} \nabla_x f(x') + \nabla_x^2 f(x')\Delta x_{nt} + Dh(x')^T\lambda \\ h(x) + Dh(x')\lambda \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Rightarrow \boxed{\begin{bmatrix} \nabla^2 f(x') & A^T \\ A & 0 \end{bmatrix}\begin{bmatrix} \Delta x_{nt} \\ \lambda \end{bmatrix} = -\begin{bmatrix} \nabla f(x') \\ Ax' - b \end{bmatrix}}$

- Introduce **residual** function $r(y) = r(x,\lambda) = \begin{bmatrix} \nabla f(x) + A^T\lambda \\ Ax - b \end{bmatrix}_{(n+p)\times 1}$
- Linearizing $r(y) = 0$ gives $r(y + \Delta y) \approx r(y) + Dr(y)\Delta y = 0$

$\boxed{\text{Intuition: } r(y^*) \approx 0}$

$$\Rightarrow \begin{bmatrix} \nabla f(x) + A^T\lambda \\ Ax - b \end{bmatrix} + Dr(y)\begin{bmatrix} \Delta x \\ \Delta\lambda \end{bmatrix} = \begin{bmatrix} \nabla f(x) + A^T\lambda \\ Ax - b \end{bmatrix} + \begin{bmatrix} \nabla^2 f(x) & A^T \\ A & 0 \end{bmatrix}\begin{bmatrix} \Delta x \\ \Delta\lambda \end{bmatrix} = 0$$

$$\Rightarrow \boxed{\begin{bmatrix} \nabla^2 f(x) & A^T \\ A & 0 \end{bmatrix}\begin{bmatrix} \Delta x_{nt} \\ \Delta\lambda_{nt} \end{bmatrix} = -\begin{bmatrix} \nabla f(x) + A^T\lambda \\ Ax - b \end{bmatrix}}$$

Δx_{nt}: Primal Newton step
$\Delta\lambda_{nt}$: Dual Newton step

$$\boxed{Dr(y) = \begin{bmatrix} D_y(\nabla f(x) + A^T\lambda) \\ D_y(Ax - b) \end{bmatrix} = \begin{bmatrix} D_x(\nabla f(x) + A^T\lambda) & D_\lambda(\nabla f(x) + A^T\lambda) \\ D_x(Ax - b) & D_\lambda(Ax - b) \end{bmatrix} = \begin{bmatrix} \nabla^2 f(x) & A^T \\ A & 0 \end{bmatrix}}$$

Infeasible start Newton's Method

- Primal-dual interpretation
  - Update both primal x and dual λ (or ν)
  - Satisfy the optimality conditions approximately $r(y) = 0$
- The Newton step $\Delta x_{nt}, \Delta \nu_{nt}$ is **not a descent direction** unless $Ax - b = 0$
  - $\frac{d}{dt} f(x + t\Delta x_{nt})|_{t=0} = Df(x)\Delta x_{nt} = \nabla f(x)^T \Delta x_{nt} = -\Delta x_{nt}^T(\nabla^2 f(x)\Delta x_{nt} + A^T w),\ \boldsymbol{w = \nu + \Delta\nu_{nt}}$
    $= -\Delta x_{nt}^T \nabla^2 f(x) \Delta x_{nt} + (Ax - b)^T w \ \not< 0 \ if\ Ax - b \neq 0$

INFEASIBLE START NEWTON METHOD

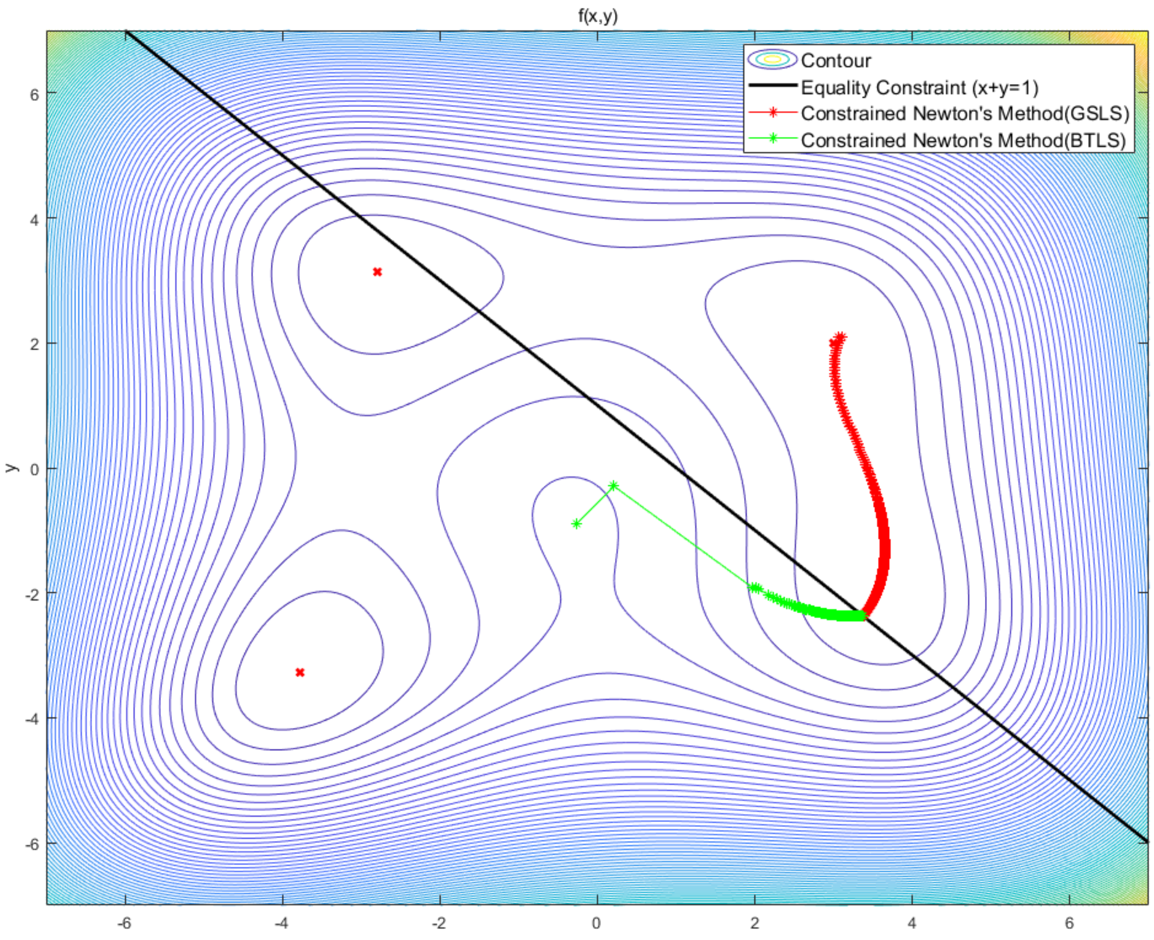
Choose initial guess x_0, *convergence tolerance* tol, $\tilde{H}_0 = I$
while $\|r(x,\nu)\|_2 \ge tol$ ***do***
  $S = \left[\tilde{H}_k \quad A^T ; A \quad 0\right]$, $P^T S P = LDL^T$
  $\left[\Delta x_{nt} ; \Delta \nu_{nt}\right] = -PL^{-T}D^{-1}L^{-1}P^T r(x,\nu)$
  Choose t *that minimizes* $\phi(\eta_k) = \|r(x + t\Delta x_{nt}, \nu + t\Delta \nu_{nt})\|_2$ *by BTLS*
    $t := 1$
    While $\|r(x + t\Delta x_{nt}, \nu + t\Delta \nu_{nt})\|_2 > (1 - \alpha t)\|r(x,\nu)\|_2 \quad t := \beta t$
  $x_{k+1} \leftarrow x_k + t\Delta x_{nt},\ \nu_{k+1} \leftarrow \nu_k + t\Delta \nu_{nt}$
  Update $\tilde{H}_k$ via BFGS
end while

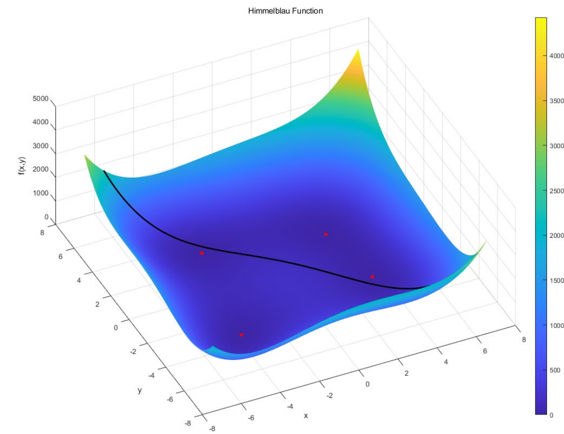
- But, the residual decreases in norm at each iteration because:

$$\frac{d}{dt}\|r(y + t\Delta y)\|_2|_{t=0} = -\|r(y)\|_2$$

Test on Himmelblau Function

- One equality constraint, $x + y = 1$






```
Infeasible start Newton Method with Golden Section Line Search using Numerical Gradients
x=3.0514, y=2.0995, norm(residual,2)=8.5377, A*x_k-b=4.1509, obj_func=0.37961
x=3.0681, y=2.0328, norm(residual,2)=7.5691, A*x_k-b=4.1009, obj_func=0.23912
x=3.0571, y=1.9954, norm(residual,2)=7.3762, A*x_k-b=4.0525, obj_func=0.1179
⋮
x=3.3723, y=-2.3723, norm(residual,2)=1.0172e-05, A*x_k-b=6.654e-06, obj_func=7.9998
x=3.3723, y=-2.3723, norm(residual,2)=1.0044e-05, A*x_k-b=6.5707e-06, obj_func=7.9998
x=3.3723, y=-2.3723, norm(residual,2)=9.9187e-06, A*x_k-b=6.4884e-06, obj_func=7.9999
```

```
Infeasible start Newton Method with Backtracking LineSearch using Numerical Gradients
x=0.20963, y=-0.29463, norm(residual,2)=27.6919, A*x_k-b=-1.085, obj_func=171.5157
x=1.9737, y=-1.9221, norm(residual,2)=63.5154, A*x_k-b=-0.94832, obj_func=83.2498
x=2.0056, y=-1.9243, norm(residual,2)=59.1042, A*x_k-b=-0.91868, obj_func=80.9082
⋮
x=3.3723, y=-2.3723, norm(residual,2)=1.0291e-05, A*x_k-b=-1.6688e-07, obj_func=8
x=3.3723, y=-2.3723, norm(residual,2)=1.0131e-05, A*x_k-b=-1.6427e-07, obj_func=8
x=3.3723, y=-2.3723, norm(residual,2)=9.9724e-06, A*x_k-b=-1.617e-07, obj_func=8
```

Contents

$$\min_x \ f(x) \ \ s.t. \ \ h(x)=0, \ g(x) \le 0$$

$$x \in R^n, \ f: R^n \to R, \ h: R^p \to R, \ g: R^m \to R$$

Barrier Method

- The barrier method solves a sequence of equality constrained problems where the inequality constraints are replaced with a so-called **barrier function** that is added to objective function.

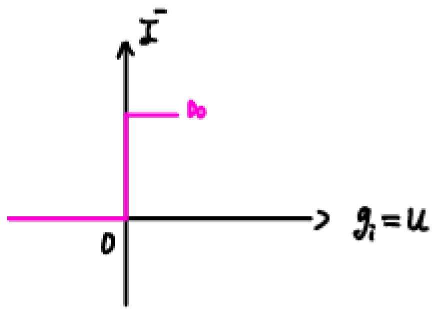
Original Problem (P)

- $\min_x f(x)$ subject to $h(x) = 0, \quad g(x) \leq 0$

Reformulated Problem

- $\min_x f(x) + \sum_{i=1}^{m} I^-(g(x))$ subject to $h(x) = 0$

  where $I^-(u) = \begin{cases} 0 & if\ u \leq 0 \\ \infty & if\ u > 0 \end{cases}$



- But, now we have a **<u>non-differentiable</u>** objective function!

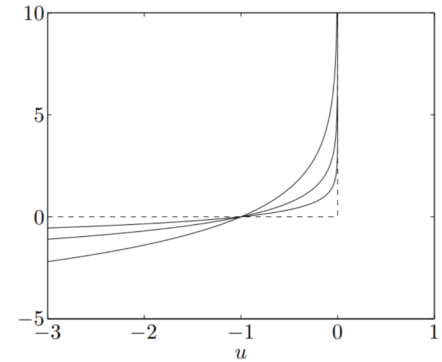
Barrier Method

- We approximate the previous representation by adding the **log barrier function**.

Centering Problem (P*)

- $\min_x f(x) - \frac{1}{\tau}\sum_{i=1}^{m} log\,(-g_i(x))$ subject to $h(x) = 0$

  where as $g_i^-(x) \to 0, -log\,(-g_i(x)) \to \infty$



- For $\tau > 0$, $\frac{1}{\tau} log(-g(x))$ is a smooth approximation of $I^-(u)$
- Approximation improves as $\tau \to \infty$. But for any value of τ, the log barrier approaches ∞ if any $g_i(x) \to 0$.
- Numerically unstable as $\tau \to \infty$.
- For sufficiently large $\tau > 0$, the solution to P*, denoted as $x^*(t)$, can be obtained by the Newton method.
- **Key idea:** Start with some small value of τ, solve P* and use that $x^*(t)$ as a hot-start for the next iteration, for which τ is increased. “Centering step”
  - Repeat until $\frac{m}{t} \le \varepsilon$ where ε is a measure of “how close you want to get to inequality constraint.”

Barrier Method

- Let $\phi(x) = -\sum_{i=1}^{m} log\ (-g_i(x))$
- Suppose we have **linear equality** and **linear inequality constraints**
  - $h(x) = Ax - b = 0,\ \ h_i(x) = a_i^T x - b_i = 0$
  - $g(x) = Cx - d \leq 0,\ \ g_i(x) = c_i^T x - d_i \leq 0$
- Then, the log barrier functions becomes
  - $\phi(x) = -\sum_{i=1}^{m} log\ (d_i - c_i^T x)$
  - $\nabla_x \phi(x) = C^T d'$ where $d_i' = \frac{1}{d_i - c_i^T x}$
  - $\nabla_x^2 \phi(x) = C^T diag\left(d'^2\right) C$ with dom $\phi = \{x | c_i^T x < d_i\}$
- Also, centering step problem P* becomes

$$\min_x\ \tilde{f}(x)\ \text{ subject to }\ Ax = b\ \ \Rightarrow\ \boxed{\min_x\ \tau f(x) + \phi(x)\ \text{ subject to }\ Ax = b}$$

Barrier Method

- $\nabla \tilde{f}(x) = \tau \nabla f(x) + \nabla \phi(x) = \tau \nabla f(x) + \underline{C^T d'}$
- $$\begin{aligned} \nabla^2 \tilde{f}(x) &= \tau D(\nabla f(x)) + D(\nabla \phi(x)) \\ &= \tau \nabla^2 f(x) + \nabla^2 \phi(x) \\ &= \tau \nabla^2 f(x) + \underline{C^T diag(d'^2) C} \end{aligned}$$

Centering Problem (P*)

$$\min_x \tilde{f}(x) = \tau f(x) + \phi(x)$$
subject to $Ax = b$

- Gradient: $\nabla f(x)$ can be obtained by numerical differentiation
- Hessian: $\nabla^2 f(x)$ can be obtained by BFGS update
- So, KKT system becomes

$$\begin{bmatrix} \nabla^2 \tilde{f}(x) & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x_{nt} \\ \Delta \lambda_{nt} \end{bmatrix} = - \begin{bmatrix} \nabla \tilde{f}(x) + A^T \lambda \\ Ax - b \end{bmatrix}$$

$$\Rightarrow \underbrace{\begin{bmatrix} \boxed{\tau \nabla^2 f(x) + C^T diag(d'^2) C} & A^T \\ A & 0 \end{bmatrix}}_{\tilde{S}} \begin{bmatrix} \Delta x_{nt} \\ \Delta \lambda_{nt} \end{bmatrix} = - \begin{bmatrix} \boxed{\tau \nabla f(x) + C^T d'} + A^T \lambda \\ Ax - b \end{bmatrix}$$

Δx_{nt}: Primal Newton step
$\Delta \lambda_{nt}$: Dual Newton step

Barrier Method

- Barrier method requires an initial point that is **strictly feasible** for all inequality constraints.

BARRIER METHOD FOR LINEAR EQUALITY AND INEQUALITY CONSTRAINTS

Choose strictly feasible x *, convergence tolerance* tol *,* $\mu > 1,\ \tau = \tau_0$

while $m/\tau \geq tol$ ***do***

  Initialize $\tilde{H}_0 = \tau I + C^T diag(d'^2)C$

  while $\left\| r(x,\nu) \right\|_2 \geq tol$ ***do***

    $\tilde{S} = \left[\tilde{H}_k \quad A^T ; A \quad 0 \right],\ \mathrm{P}^T \tilde{S} P = LDL^T$

    $\left[\Delta x_{nt} ; \Delta \nu_{nt} \right] = -PL^{-T} D^{-1} L^{-1} P^T r(x,\nu)$

    Choose t *that minimizes* $\phi(\eta_k) = \left\| r(x + t\Delta x_{nt}, \nu + t\Delta \nu_{nt}) \right\|_2$ *by BTLS*

    $x_{k+1} \leftarrow x_k + t\Delta x_{nt},\ \nu_{k+1} \leftarrow \nu_k + t\Delta \nu_{nt}$

    $r(x,\nu) \leftarrow \left[\tau \nabla f(x_{k+1}) + C^T d'_{k+1} + A^T \nu_{k+1}; \quad Ax_{k+1} - b \right]$

    Update $\tilde{H}_k$ *via BFGS*

  end while

  $x \leftarrow x^*(t)$

  $\tau \leftarrow \mu\tau$

end while

"Centering step"
Compute $x^*(t)$ by minimizing $\tau f(x) + \phi(x)$ subject to $Ax = b$

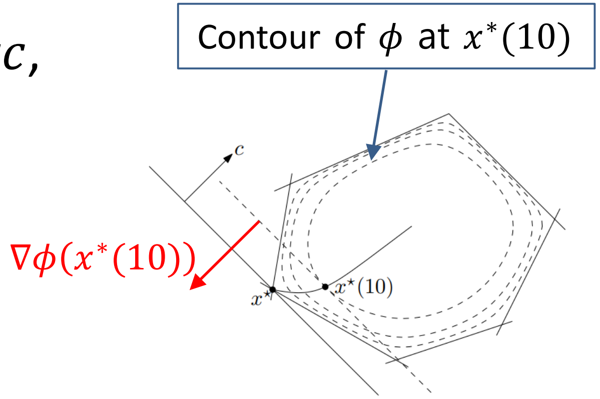
Barrier Method on Inequality-constrained LP

- Example on **inequality-constrained LP**
- $\min_{x} \ \tau c^T x - \sum_{i=1}^{m} \log\left(d_i - c_i^T x\right), f(x) = c^T x$
- The barrier function corresponds to polyhedral constraint $Cx - d \leq 0$
- The KKT system, or the optimality conditions $(\nabla_x L(x, \cancel{\lambda}) = 0 \quad \because A = 0)$

$$\nabla_x(\tau c^T x) + \nabla_x\left(\phi(x)\right) = 0 \ \Rightarrow \tau c + C^T d' = 0$$

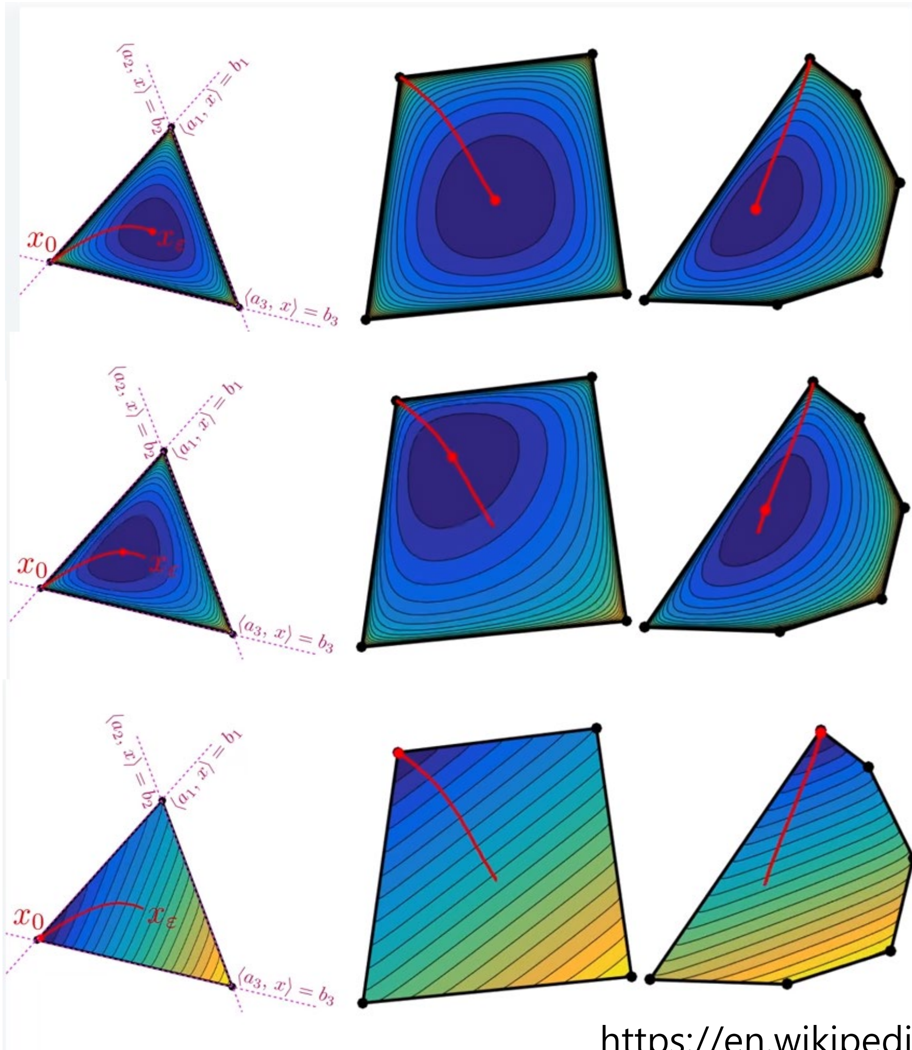
- **Geometric Interpretation**: gradient $\nabla\phi(x^*(t)) = -\tau c$,
  - must be parallel to $-c$
  - Hyperplane $\{x|\ c^T x = c^T x^*(t)\}$ lies tangent

  to contour of ϕ at $x^*(t)$



(From B & V page 565)

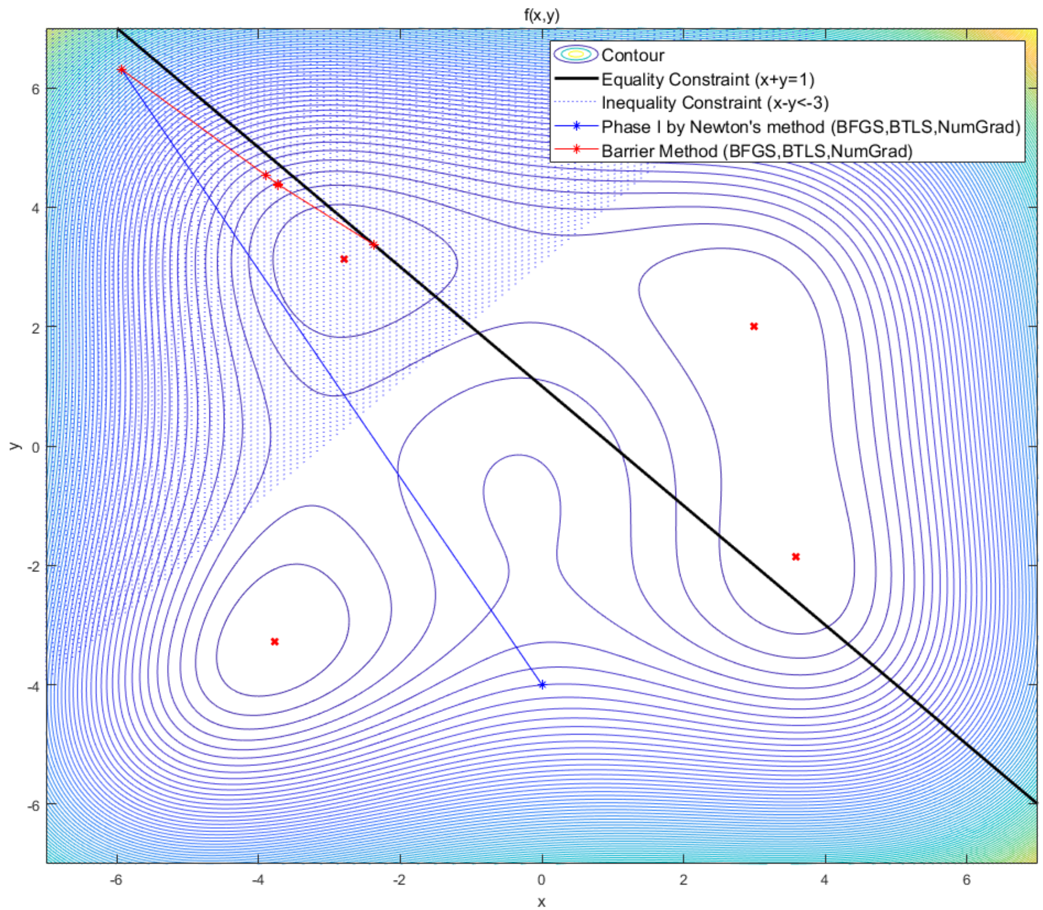
Barrier Method on Inequality-constrained LP



https://en.wikipedia.org/wiki/Interior-point_method

Test on Himmelblau Function

- One equality constraint: $x + y = 1$
- One inequality constraint: $x - y \leq -3$




```
(Phase I) Barrier method with Basic Backtracking Line Search using Numerical Gradients

pk =

     0
    -4
    14
     1

t=1, x=1.1786, y=-2.6786, s=13.5714, A*x_k-b=-2.5, phase1_objfunc=11.6672
t=1, x=1.6796, y=-1.9296, s=12.915, A*x_k-b=-1.25, phase1_objfunc=11.0735
t=1, x=-5.9371, y=6.3121, s=-25.0452, A*x_k-b=-0.625, phase1_objfunc=-27.8049-3.14159i
Strictly feasible solution is found, s=-25.0452
```

```
t=5, x=-4.5636, y=4.9399, A*x_k-b=-0.62378, obj_func=382.8798
t=5, x=-4.5561, y=4.9348, A*x_k-b=-0.62134, obj_func=379.6278
t=5, x=-4.5506, y=4.9317, A*x_k-b=-0.61892, obj_func=377.4118
⋮
t=5000, x=-2.3723, y=3.3723, A*x_k-b=-9.9907e-12, obj_func=8
t=5000, x=-2.3723, y=3.3723, A*x_k-b=-4.9956e-12, obj_func=8
t=5000, x=-2.3723, y=3.3723, A*x_k-b=-2.4976e-12, obj_func=8
```

Why Barrier Method?

- Strengths
  - Polynomial complexity in the worst case LP
    - Combinatorial complexity for Simplex method
  - Viable linear algebra operation
    - IPM does only solving linear system, which is straightforward
    - Suitable for large, sparse problems
  - Robust to “scaling” of problem
    - Can handle large-scale problems
- Weaknesses
  - Each centering step is an expensive operation
  - Converges to a local minimum if problem is not convex
    - Interior-point methods for nonconvex nonlinear programming have been developed by Benson, Shanno, and Vanderbei in 2000.
    - Otherwise, use global optimizers such as CEALM

Contents

Barrier Method MATLAB Code Implementation

- Define Constrained Parameter Optimization Problem

```
N = 200; % Number of collocation nodes
tf = 10; % final time fixed problem
h_step = tf/N;
Waypoint = false;
WP = [2,150;5,50]; % for zk

% Define Decision Variables, X (3*N+3)
n_var = 3; % [z,v,u]
n = 3*N+3; % dimension of X
x = zeros(n,1); % column [z0,v0,u0,...,zN,vN,uN], zk = k*n_var+1; vk = k*n_var+2; uk = k*n_var+3, k=0,1.,...,N

% Define Equality Constraints (AX=b or AX-b=0, (N+N+2+2)-by-n)
% 1. Dynamic Constraints (N+N)
A_dyn = zeros(2*N,n);
for k=1:N
    A_dyn(k,[(k-1)*n_var+1, k*n_var+1, (k-1)*n_var+2, k*n_var+2, (k-1)*n_var+3, k*n_var+3]) = [-1, 1, -h_step/2, -h_step/2, h_step^2/12, -h_step^2/12];
    A_dyn(k+N,[(k-1)*n_var+2, k*n_var+2, (k-1)*n_var+3, k*n_var+3]) = [-1, 1, h_step/2, h_step/2]; % k=1일때, v0,v1,u0,u1
end

% 2. Initial and Final Conditions
A_cond = zeros(2+2,n);
k=1; A_cond(1,(k-1)*n_var+1) = 1; A_cond(2,(k-1)*n_var+2) = 1; % z0,v0
k=N; A_cond(3,    k*n_var+1) = 1; A_cond(4,k*n_var+2) = 1;      % zN,vN

% 3. Form matrix A
A = [A_dyn;A_cond];

% 4. Form RHS matrix b, (N+N+2+2)-by-1
b = zeros(2*N+4,1); % column
for k=1:N
    b(k+N) = g*h_step;
end
b([2*N+1,2*N+2,2*N+3,2*N+4]) = [100;10;0;0];
```

Barrier Method MATLAB Code Implementation

- Define Constrained Parameter Optimization Problem

```
% Define Inequality Constraints (Cx<=d or Cx-d<=0, 2*N+2)
% 1. Upper and Lower bounds on variables
C = [eye(n);-eye(n)];
d = 1e3*ones(2*n,1); % column

% Define Objective Function
% 1.objfunc = (u*u')*tf/(2*N);
% 2.objfunc = (u(1:end-1)*u(1:end-1)' + u(1:end-1)*u(2:end)' + u(2:end)*u(2:end)')*tf/(3*N)/2;
%obj_func = @(x) sum(x(n_var:3:n,:).^2,1)*tf/(2*N); % u = x(n_var:3:n); % column
obj_func = @(x) (sum(x(n_var:3:n-n_var,:).^2,1) + sum(x(n_var:3:n-n_var,:).*x(2*n_var:3:n,:),1) + sum(x(2*n_var:3:n,:).^2,1) )*tf/(3*N)/2;
obj1 = false; %true if obj_func1 is used

% Define Waypoint (if any) and add constraints
if Waypoint
    sz = size(WP);
    A_wp = zeros(sz(2),n);
    b_wp = zeros(sz(2),1);
    for i=1:sz(2)
        ti = WP(i,1); % time
        idx = ti/(tf/N); % index
        A_wp(i,idx*n_var+1) = 1;
        b_wp(i) = WP(i,2);
    end
    A = [A;A_wp];
    b = [b;b_wp];
    % for logging
    trace_data_wp = zeros(100,4);
else
    trace_data = zeros(100,4);
end
```

Barrier Method MATLAB Code Implementation

- Solving by Barrier Method

```
x0   = ones(n,1);

x_k  = x0;
p    = length(b); % number of equality constraints
m    = length(d); % number of inequality constraints
H_k  = eye(n); I = eye(n);
nu_k = zeros(p,1);
t0   = 10; % Initialize centering step
pk   = [x_k;t0];
BM_obj = @Project_BM_objfunc;

g_k = num_grad(BM_obj, h, pk)';
% Check with analytic gradient
d_prime = 1./(d-C*x_k);
g_k_check = t0*num_grad(obj_func,h,x_k)' + C'*d_prime;
assert(norm(g_k-g_k_check,2)<1e-4,"numerical gradient very different from analytical gradient for BM")

yk = [x_k; nu_k;g_k]; % (n)+(p)+(n)
residual_func = @(yk) [yk(n+p+1:n+p+n)+A'*yk(n+1:n+p); A*yk(1:n)-b]; % equiv to @(x_k,nu_k,g_k) [g_k+A'*nu_k;A*x_k-b]
residual = residual_func(yk);
```

Barrier Method MATLAB Code Implementation

- Solving by Barrier Method

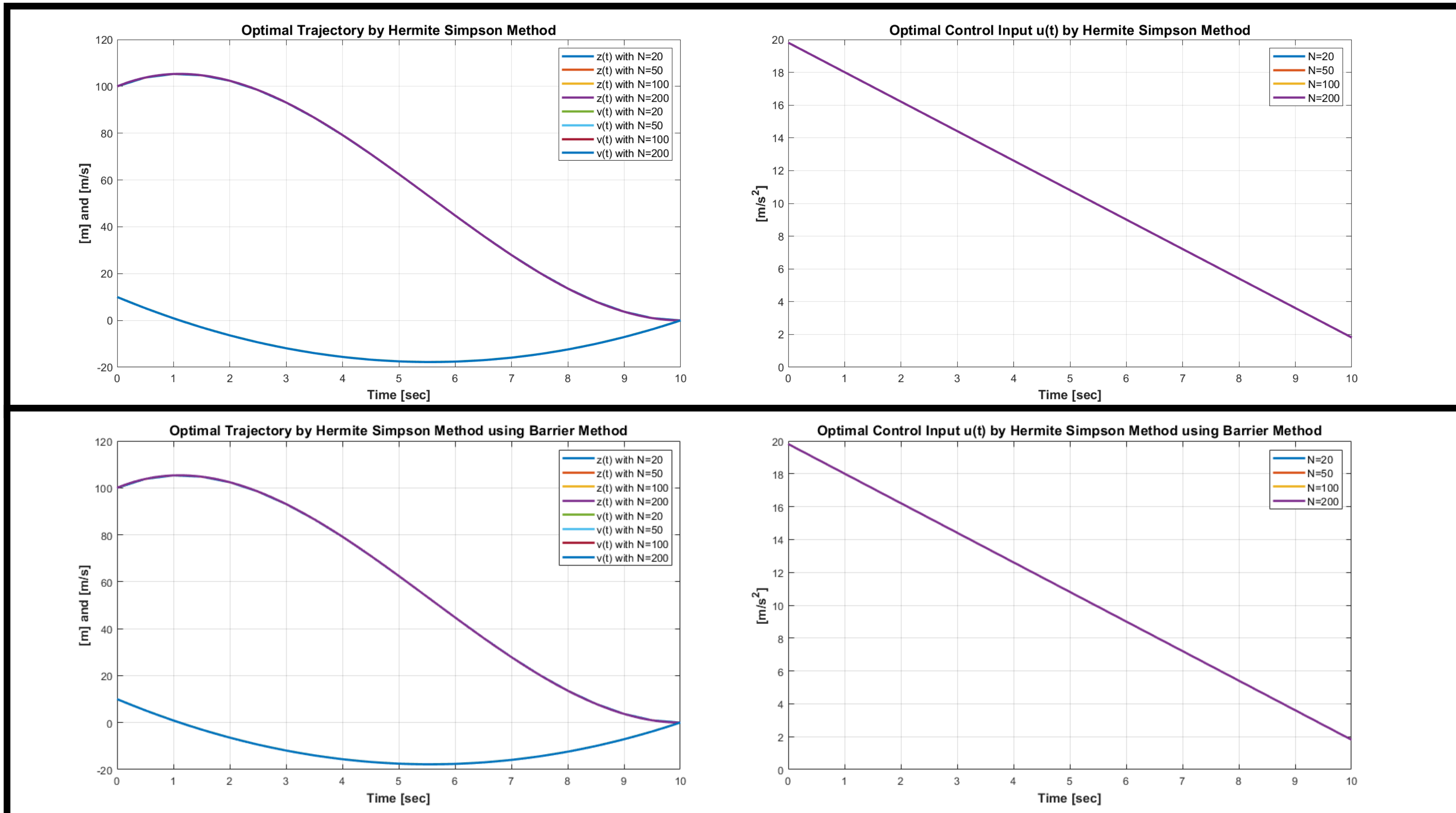
```
mu = 10; maxiter=1e2; j=1; done = false;
while m/pk(n+1) > BM_tol && ~done %pk(n+1) = t0
    %pk
    k=1;
    d_prime = 1./(d-C*pk(1:n)); %pk(1:n) = x
    H_k = pk(n+1)*I + C'*diag(d_prime.^2)*C; % initialize Hessian with analytic results
    while norm(residual,2) > conv_tol
        if  k > maxiter
            break;
        end
        if j>7 && norm(A*pk(1:n)-b,2) < h*1e-1 % j counts the magnitude of pk(n+1) = t or tau
           done = true;
           break;
        end
        % 1. Compute KKT newton_step using current x_k, H_k by LDL' factorisation
        S = [H_k, A';A, zeros(p,p)];
        [L,D,P] = ldl(S);
        newton_step = -P*(L^-1)'*(D^-1)*(L^-1)*P' * residual;
        pn_step = newton_step(1:n);
        dn_step = newton_step(n+1:n+p);
        % 2. Backtracking Line Search on Residual
        t = BacktrackingLineSearch_BFGS_EQ(residual_func, yk, -residual, n, p);
        % 3. Update x_k, nu_k, g_k, yk, residual using pk
        pk(1:n) = pk(1:n) + t*pn_step;
        nu_k = nu_k + t*dn_step;
        g_k = num_grad(BM_obj,h,pk)';
        yk = [pk(1:n);nu_k;g_k];
        residual = residual_func([pk(1:n);nu_k;g_k]);
        % 4. Update Hessian H_k with BFGS (not inverse)
        y_k = num_grad(BM_obj,h,[pk(1:n)+t*pn_step;pk(n+1)])' - num_grad(BM_obj,h,pk)';
        s_k = t*pn_step;
        H_k = H_k - ((H_k*s_k)*s_k'*H_k')/(s_k'*H_k*s_k) + (y_k*y_k')/(y_k'*s_k);
        %z = pk(1:n_var:n)
        %v = pk(2:n_var:n)
        %u = pk(3:n_var:n)
        % Save log
        trace_data((j-1)*100+k,1) = pk(n+1);
        trace_data((j-1)*100+k,2) = norm(residual,2);
        trace_data((j-1)*100+k,3) = norm(A*pk(1:n)-b,2);
        trace_data((j-1)*100+k,4) = obj_func(pk(1:n));
        text = ['t=', num2str(pk(n+1)), ', norm(residual,2)=', num2str(norm(residual,2)), ', norm(A*x_k-b,2)=', num2str(norm(A*pk(1:n)-b,2)), ', obj_func=', num2str(obj_func(pk(1:n)))];
        disp(text)
        % check if BM_objfunc blows up
        assert(isreal(BM_obj(pk)),'BM_objfunc blows up! Increase initial t0')
        k=k+1;
    end

    % check norm(residual,2)
    if norm(residual,2) < conv_tol
        disp('norm(residual,2) is smaller than convergence tolerance');
        break;
    else
        % Update t0
        disp('Updating t0 and starting again...')
        pk(n+1) = mu*pk(n+1);
        j=j+1;
    end


end
```

BARRIER METHOD FOR LINEAR EQUALITY AND INEQUALITY CONSTRAINTS

Choose strictly feasible x, *convergence tolerance* tol, $\mu > 1$, $\tau = \tau_0$
while $m/\tau \ge tol$ **do**
  Initialize $\tilde{H}_0 = \tau I + C^T diag(d'^2) C$
  while $\|r(x,\nu)\|_2 \ge tol$ **do**
    $\tilde{S} = \begin{bmatrix} \tilde{H}_k & A^T ; A & 0 \end{bmatrix}$, $P^T \tilde{S} P = LDL^T$
    $[\Delta x_{nt}; \Delta\nu_{nt}] = -PL^{-T} D^{-1} L^{-1} P^T r(x,\nu)$
    Choose t *that minimizes* $\phi(\eta_k) = \|r(x + t\Delta x_{nt}, \nu + t\Delta\nu_{nt})\|_2$ *by BTLS*
    $x_{k+1} \leftarrow x_k + t\Delta x_{nt}$, $\nu_{k+1} \leftarrow \nu_k + t\Delta\nu_{nt}$
    $r(x,\nu) \leftarrow \begin{bmatrix} \tau\nabla f(x_{k+1}) + C^T d'_{k+1} + A^T \nu_{k+1}; & Ax_{k+1} - b \end{bmatrix}$
    Update $\tilde{H}_k$ *via BFGS*
  end while
  $x \leftarrow x^*(t)$
  $\tau \leftarrow \mu\tau$
end while

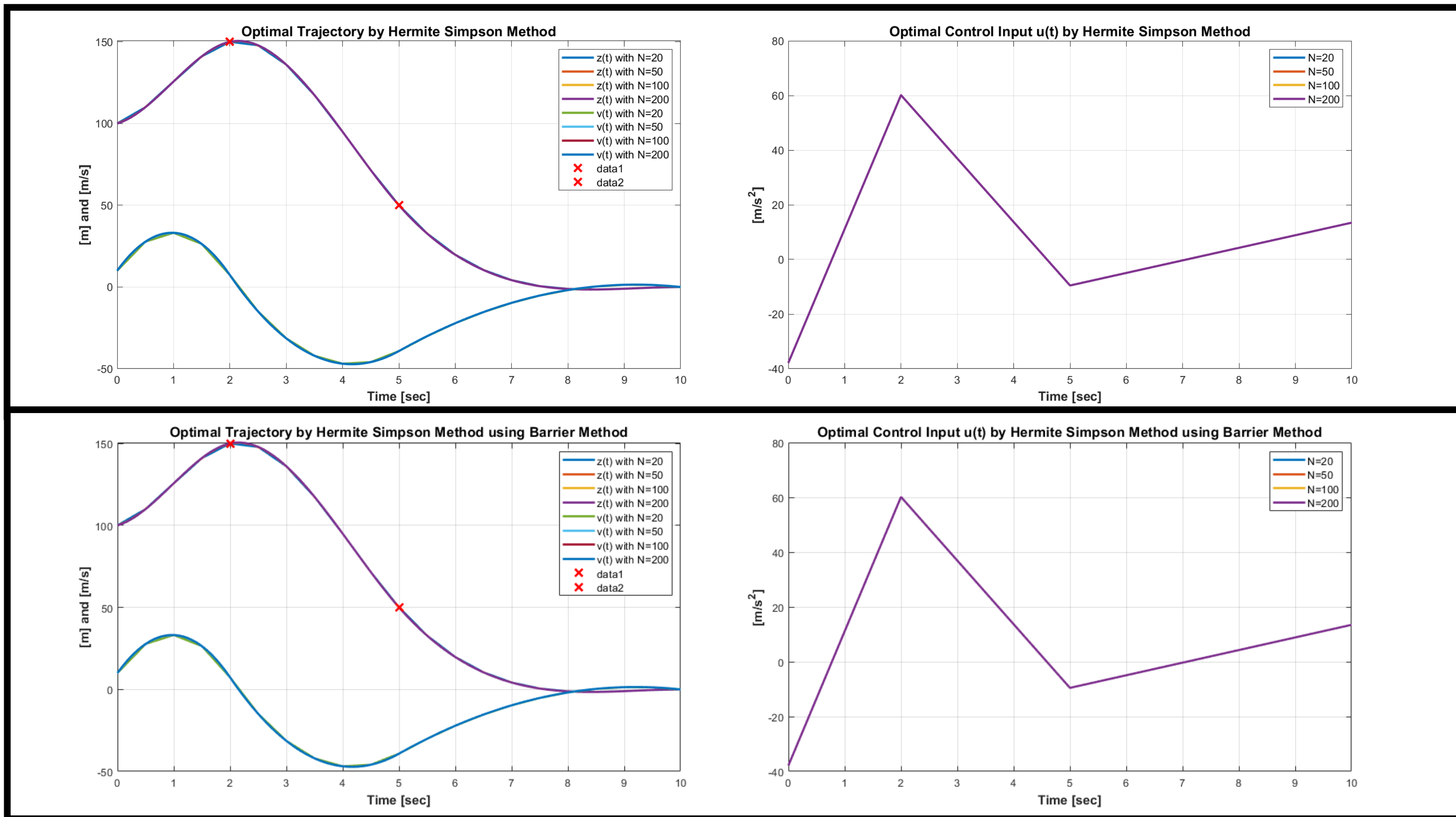
Comparison with MATLAB's *fmincon*

- **Without** waypoint constraints: Objective function = 719.2805



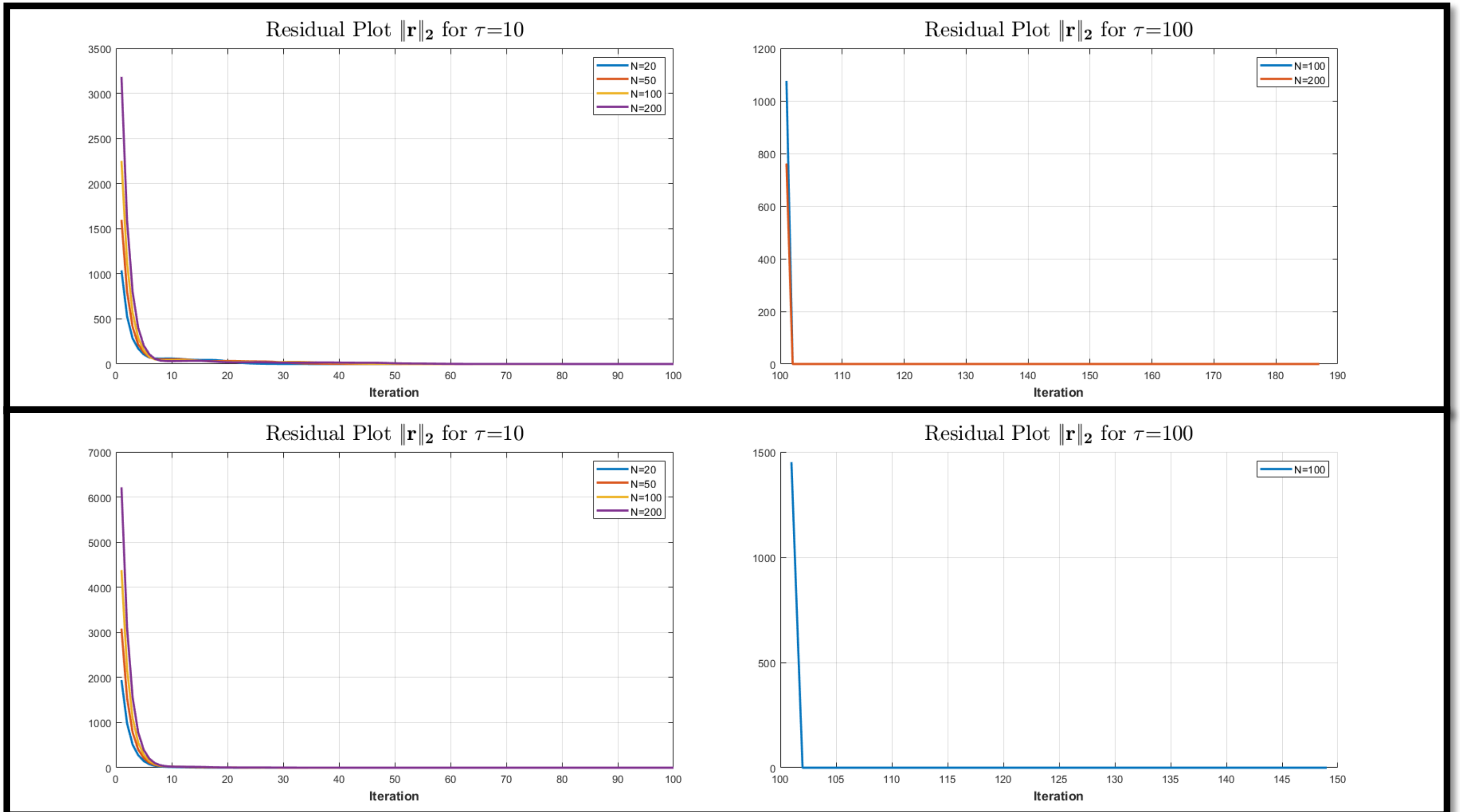
Comparison with MATLAB's *fmincon*

- **With** waypoint constraints: Objective function = 2612.058 $z(2) = 150, z(5) = 50$



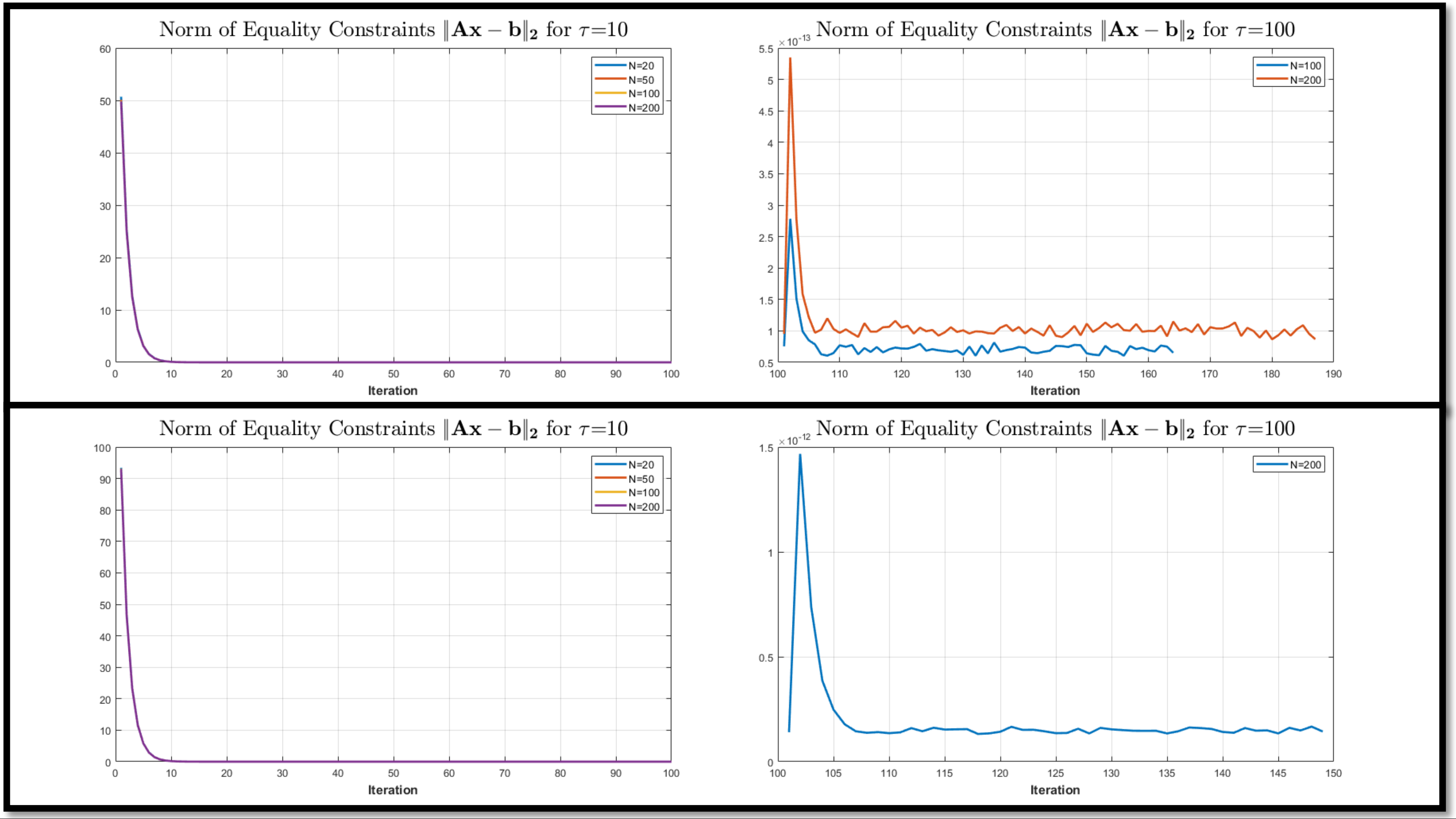
Barrier Method Results

- Residual Plot (without waypoints, with waypoints)



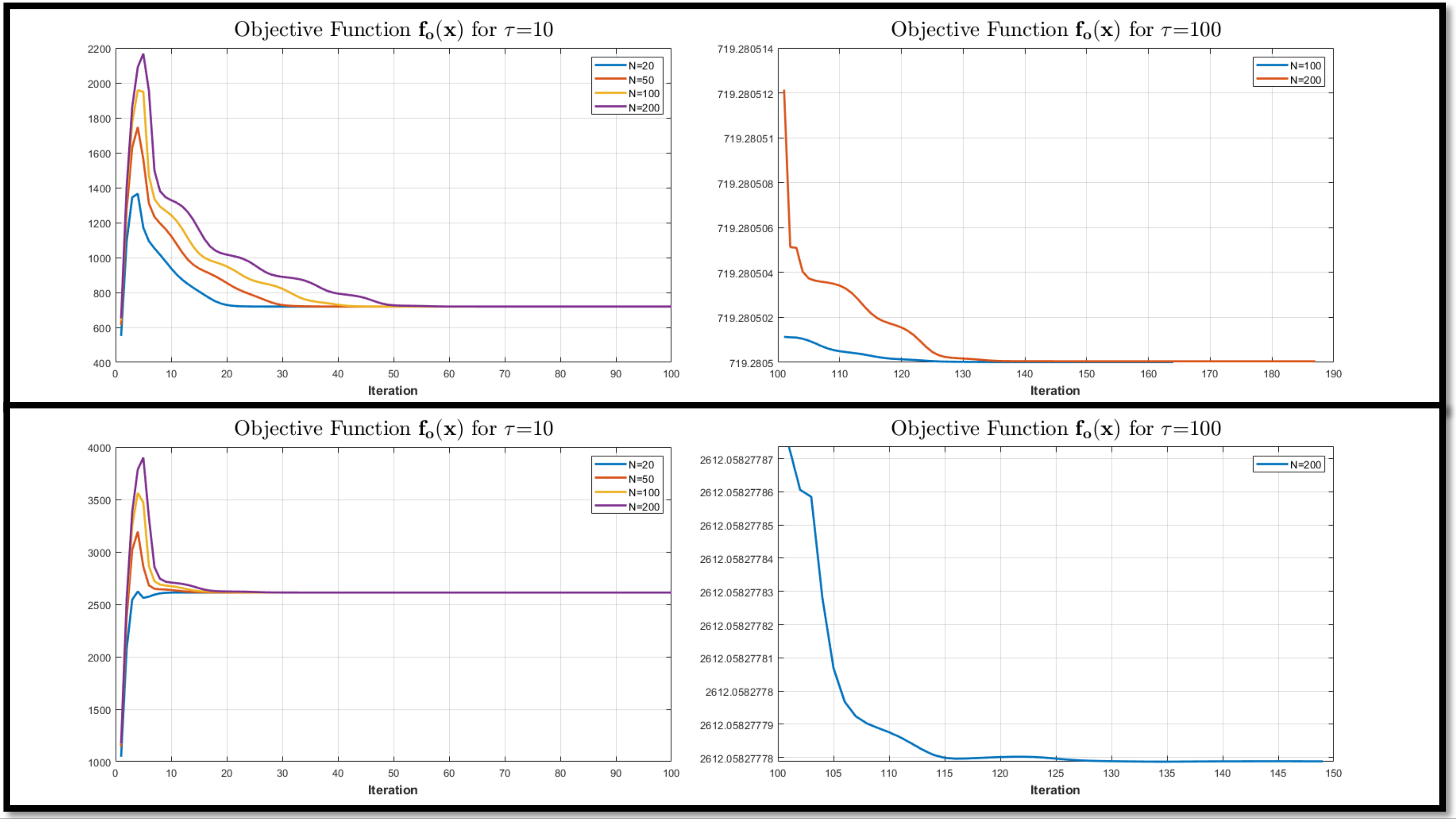
Barrier Method Results

- Norm of Equality Constraints Plot (without waypoints, with waypoints)



Barrier Method Results

- Objective Function Plot (without waypoints, with waypoints)



Contents

Conclusion

- Implemented MATLAB Code for parameter optimization algorithms
  - Unconstrained Problem
    - Steepest Descent
    - Newton Method
    - Quasi-Newton Method (BFGS)
  - Equality Constrained Problem
    - Constrained Newton Method (KKT)
    - Infeasible start Newton Method
  - Inequality Constrained Problem
    - Barrier Method
    - Phase I Optimization Problem
- Solved optimal guidance problem using MATLAB's *fmincon* and Barrier method
  - Additional waypoint constraints at two points
  - Comparison of Results

Reflection

- Had a chance to manually code various optimization algorithms
  - ➢ Great experience to understand how the algorithm works
- Gained skills to code algorithms ***independently***
  - ➢ Should be prepared if such a need arises, possibly in near future

Thank you

SAI
STRATEGIC
AEROSPACE
INITIATIVE

References

- Boyd, S., & Vandenberghe, L. (2004). *Convex Optimization*. Cambridge: Cambridge University Press. doi:10.1017/CBO9780511804441
- Nocedal, Jorge., & Wright. S. (2006). *Numerical Optimization*. Spriner-Verlag New York. Doi: 10.1007/978-0-387-40065-5
- Daniel, Robinson P. “Line-Search Methods for Smooth Unconstrained Optimization.” *Department of Applied Mathematics and Statistics Johns Hopkins University*, 17 Sept. 2020, www.ams.jhu.edu/~abasu9/AMS_553-761/lecture05_handout2.pdf.
- Min Jea, Tahk. “Part B: Parameter Optimization.” *Department of Aerospace Engineering, KAIST*

Appendix

Phase I Optimization Problem

- The Barrier method requires an initial point that is strictly feasible for all inequality constraints.

<Phase I optimization problem>

method, i.e. $\phi(x,s) = -\sum \log(s - f_i(x))$ and form

$$\boxed{\begin{array}{ll} \underset{(x,s)}{\text{minimize}} & ts + \phi(x,s) \\ \text{s.t.} & A'x' = b \end{array}}$$

$x \in R^n,\ s \in R \Rightarrow (x,s) \in R^{n+1}$

$$\underset{m\times(n+1)}{\begin{bmatrix} A & \begin{matrix} 0 \\ \vdots \\ 0 \end{matrix} \end{bmatrix}} \underset{(n+1)\times 1}{\begin{bmatrix} x \\ s \end{bmatrix}} = \underset{m\times 1}{b}$$

$$\begin{array}{lll} \underset{(x,s)}{\text{minimize}} & s & \\ \text{s.t.} & f_i(x) \le s, \; i = 1, \dots, m & (11.19) \\ & Ax = b & \end{array}$$

- This is always strictly feasible $\because$ pick x^0 and choose s as any number larger than $\max_{i=1\dots m} f_i(x^0)$.
- Solved by barrier method, i.e. $\phi(x,s) = -\sum \log(s - f_i(x))$ and form

$d' = \frac{1}{d_i - c_i^T x}$

$$\boxed{\begin{array}{ll} \text{minimize} & \underbrace{ts + \phi(x,s)}_{\tilde{f}_0(x)} \\ \text{s.t.} & A'x' = b \end{array}}$$

$x \in R^n,\ s \in R \Rightarrow (x,s) \in R^{n+1}$

$$\underset{m\times(n+1)}{\begin{bmatrix} A & \begin{matrix} 0 \\ \vdots \\ 0 \end{matrix} \end{bmatrix}} \underset{(n+1)\times 1}{\begin{bmatrix} x \\ s \end{bmatrix}} = \underset{m\times 1}{b}$$

If $f_i(x)$ linear, i.e. $f_i(x) \le 0 \Rightarrow Cx - d \le 0$, then $\phi(x,s) = -\sum_i \log(s - c_i^T x + d_i)$, and

$$\nabla \tilde{f}_0(x) = \begin{bmatrix} \nabla_x \tilde{f}_0(x) \\ \nabla_s \tilde{f}_0(x) \end{bmatrix} = \begin{bmatrix} \nabla_x \phi(x) \\ t + \nabla_s \phi(x) \end{bmatrix}$$

$$\nabla_x \phi = \sum_{i=1}^{m} \frac{c_i}{s - c_i^T x + d_i} = \boxed{C^T \tilde{d}} \text{ where } \tilde{d}_i = \frac{1}{s - c_i^T x + d_i}, \; i = 1 \dots m$$

$$\nabla_s \phi = \sum_{i=1}^{m} \frac{-1}{s - c_i^T x + d_i} = \boxed{-\mathbf{1}^T \tilde{d}}$$

- If $s < 0$ for some (x,s), then x is strictly feasible. STOP
- If $s^* > 0$, no feasible soln exists.
- If $s^* = 0$, then x^* is feasible but cannot be used.
  $\because$ This x^* is on the boundary, so the barrier function blows up immediately.

- Note that we do not solve this problem to high accuracy. Once a feasible (x,s) with $s<0$ is found, we terminate early.

Convex Optimization by S. Boyd pg. 579

11.4 Feasibility and phase I methods

The barrier method requires a strictly feasible starting point $x^{(0)}$. When such a point is not known, the barrier method is preceded by a preliminary stage, called *phase I*, in which a strictly feasible point is computed (or the constraints are found to be infeasible). The strictly feasible point found during phase I is then used as the starting point for the barrier method, which is called the *phase II* stage. In this section we describe several phase I methods.

11.4.1 Basic phase I method

We consider a set of inequalities and equalities in the variables $x \in \mathbf{R}^n$,

$$f_i(x) \le 0, \quad i = 1, \ldots, m, \qquad Ax = b, \tag{11.18}$$

where $f_i : \mathbf{R}^n \to \mathbf{R}$ are convex, with continuous second derivatives. We assume that we are given a point $x^{(0)} \in \mathbf{dom}\, f_1 \cap \cdots \cap \mathbf{dom}\, f_m$, with $Ax^{(0)} = b$.

Our goal is to find a strictly feasible solution of these inequalities and equalities, or determine that none exists. To do this we form the following optimization problem:

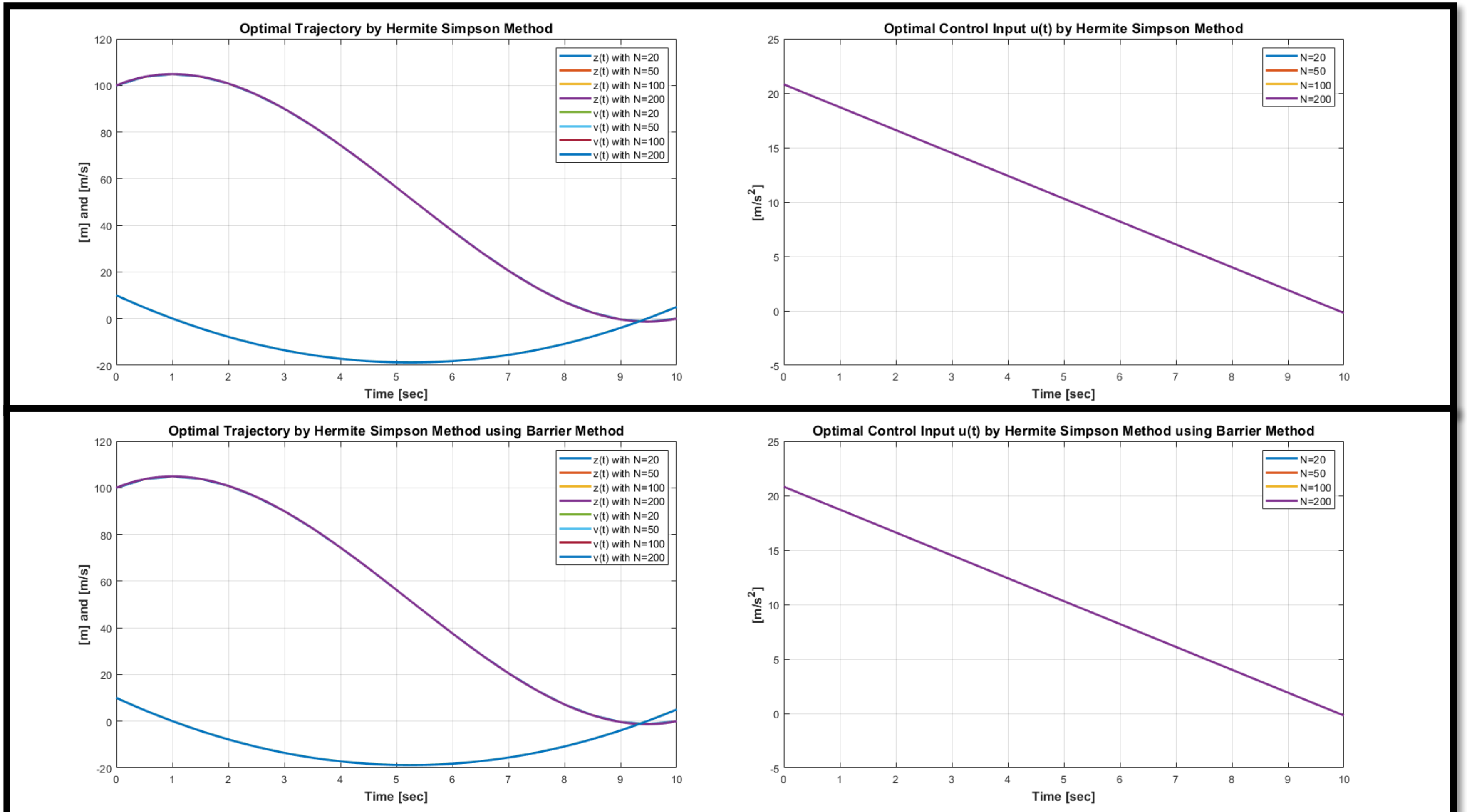
$$\begin{array}{ll} \text{minimize} & s \\ \text{subject to} & f_i(x) \le s, \quad i = 1, \ldots, m \\ & Ax = b \end{array} \tag{11.19}$$

in the variables $x \in \mathbf{R}^n$, $s \in \mathbf{R}$. The variable s can be interpreted as a bound on the maximum infeasibility of the inequalities; the goal is to drive the maximum infeasibility below zero.

This problem is always strictly feasible, since we can choose $x^{(0)}$ as starting point for x, and for s, we can choose any number larger than $\max_{i=1,\ldots,m} f_i(x^{(0)})$. We can therefore apply the barrier method to solve the problem (11.19), which is called the *phase I optimization problem* associated with the inequality and equality system (11.19).

Changing Final Time Constraints

- Tried with a problem with $v(t_f) = 5$, objective function value = 715.2305



Saturated Control Input

- For control input, set $-15 \leq u(t) \leq 15$
- **Very difficult** to converge to the optimal solution
  - ➢ Current plain backtracking line search gives step size of order 10^{-7}
  - ➢ After relaxing convergence tolerance, managed to obtain solution for N=50
  - ➢ Difficulty increases with number of collocation nodes
- Different strategy for line search algorithm is **required**
  - ➢ Many variants of backtracking line search have been studied
    - Wolfe conditions (suitable for both quasi-newton and conjugate gradient)
    - Goldstein conditions (not suitable for quasi-newton)
    - ….
- Instead of Barrier method, Primal-dual method is another option.
  - ➢ Directly solves the perturbed KKT system
  - ➢ Primal-dual generally has faster convergence than barrier

Saturated Control Input

- For control input, set $-15 \le u(t) \le 15$