Hongyi Zheng's solutions to problem 3

November 7, 2022

1 Hongyi Zheng's solutions to problem 3

[334]:

```
import itertools
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import FormatStrFormatter
```

[347]:

```
def c(X):
    return 1 + np.sqrt(4 - 3 * X[:, 0] - X[:, 1])

def objective_function(X):
    speed = c(X)
    avg_speed = (np.concatenate(([3], speed)) + np.concatenate((speed,
                                                               [1]))) / 2
    dist = np.linalg.norm(np.concatenate([X, [[1, 1]]]) - np.concatenate([[[0,␣
 ↪0]], X]), axis=1)
    return np.sum(dist / avg_speed)

def get_grad(X):
    grad = np.zeros_like(X)
    all_coords = np.concatenate([[[0, 0]], X, [[1, 1]]])
    for i in range(0, X.shape[0]):

        denom = 1 + (np.sqrt(4 - 3 * X[i, 0] - X[i, 1]) +
                     np.sqrt(4 - 3 * all_coords[i, 0] - all_coords[i, 1])) / 2
        dist = np.linalg.norm(X[i, :] - all_coords[i, :])
        grad[i, :] += ((X[i, :] - all_coords[i, :]) / dist * denom - 0.25 *
                       np.array([-3, -1]) * dist / np.sqrt(4 - 3 * X[i, 0] -␣
 ↪X[i, 1])) / denom ** 2

        denom = 1 + (np.sqrt(4 - 3 * X[i, 0] - X[i, 1]) +
                     np.sqrt(4 - 3 * all_coords[i + 2, 0] - all_coords[i + 2,␣
 ↪1])) / 2
        dist = np.linalg.norm(X[i, :] - all_coords[i + 2, :])
        grad[i, :] += ((X[i, :] - all_coords[i + 2, :]) / dist * denom - 0.25 *
                       np.array([-3, -1]) * dist / np.sqrt(4 - 3 * X[i, 0] -␣
 ↪all_coords[i + 2, 1])) / denom ** 2
```

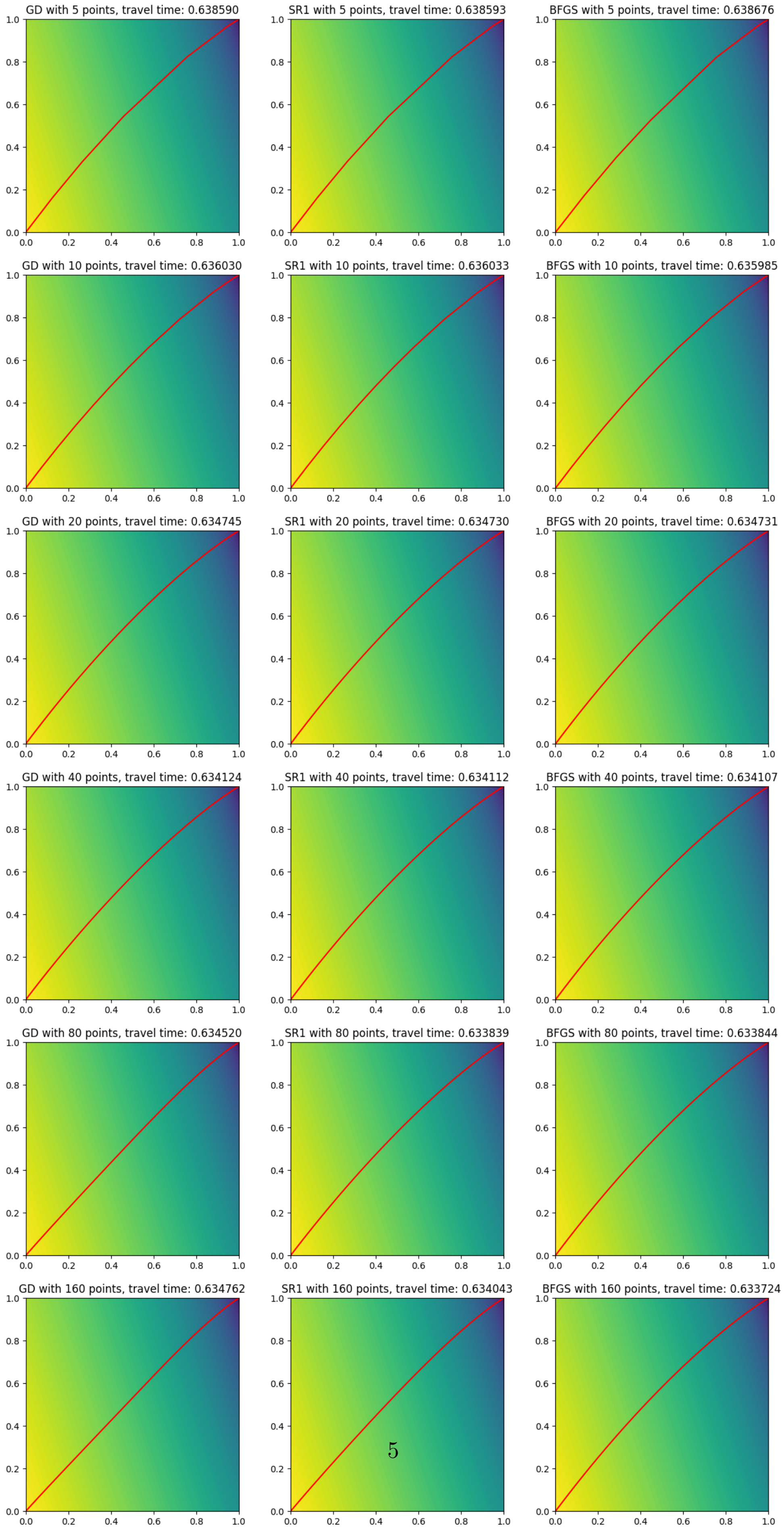
```
    return grad


def GD(X):
    step_magnitudes = []
    while True:
        alpha = 1.0
        curr_grad = get_grad(X)

        while True:
            X_next = X - alpha * curr_grad
            obj_next = objective_function(X_next)
            obj_curr = objective_function(X)
            if obj_next < obj_curr or alpha < 1e-10:
                break
            alpha /= 2

        if abs(obj_next - obj_curr) < 1e-10:
            break

        step_magnitudes.append(np.linalg.norm(X_next - X))
        X = X_next

    return X, step_magnitudes

def SR1(X):
    step_magnitudes = []
    H_k = np.eye(X.size)
    while True:
        alpha = 1.0
        curr_grad = get_grad(X)
        p_k = -H_k @ curr_grad.flatten()

        while True:
            s_k = alpha * p_k
            X_next = X + s_k.reshape(X.shape)
            obj_next = objective_function(X_next)
            obj_curr = objective_function(X)
            if obj_next < obj_curr or abs(alpha) < 1e-10:
                break
            alpha /= 2

        if abs(obj_next - obj_curr) < 1e-10:
            break

        step_magnitudes.append(np.linalg.norm(X_next - X))
        X = X_next
        next_grad = get_grad(X)
        s_k = s_k.reshape(-1, 1)
        y_k = (next_grad - curr_grad).reshape(-1, 1)
        H_k = H_k + (s_k - H_k @ y_k) @ (s_k - H_k @ y_k).T / (s_k - H_k @ y_k).
 ↪T @ y_k

    return X, step_magnitudes

def BFGS(X):
    step_magnitudes = []
    H_k = np.eye(X.size)
    while True:
        alpha = 1.0
        curr_grad = get_grad(X)
        p_k = -H_k @ curr_grad.flatten()
        while True:
            s_k = alpha * p_k
            X_next = X + s_k.reshape(X.shape)
            obj_next = objective_function(X_next)
            obj_curr = objective_function(X)

            if obj_next < obj_curr or alpha < 1e-10:
                break
            alpha /= 2

        if abs(obj_next - obj_curr) < 1e-10:
            break

        step_magnitudes.append(np.linalg.norm(X_next - X))

        X = X_next
        next_grad = get_grad(X)
        s_k = s_k.reshape(-1, 1)
        y_k = (next_grad - curr_grad).reshape(-1, 1)
        H_k = H_k + (s_k.T @ y_k + y_k.T @ H_k @ y_k) * (s_k @ s_k.T) / (s_k.T␣
 ↪@ y_k) ** 2 - (H_k @ y_k @ s_k.T + s_k @ y_k.T @ H_k) / (s_k.T @ y_k)

    return X, step_magnitudes
```

[348]:

```
fig, axs = plt.subplots(6, 3, figsize=(15, 30))

Xs, Ys = np.meshgrid(np.linspace(0, 1, 100), np.linspace(0, 1, 100))
Zs = 1 + np.sqrt(4 - 3 * Xs - Ys)

results = {fn: [] for fn in [GD, SR1, BFGS]}
```

```python
for (n, fn), ax in zip(itertools.product([5, 10, 20, 40, 80, 160], [GD, SR1, BFGS]), axs.ravel()):
    X = np.repeat(np.linspace(0, 1, n + 2)[None, 1:-1], 2, axis=0).T
    X_final, step_magnitudes = fn(X)

    travel_time = objective_function(X_final)

    all_coords = np.concatenate([[[0, 0]], X_final, [[1, 1]]])

    ax.contourf(Xs, Ys, Zs, levels=100)
    ax.plot(all_coords[:, 0], all_coords[:, 1], 'r')

    ax.margins(x=0, y=0)
    ax.set_aspect('equal')
    ax.set_title(f'{fn.__name__} with {n} points, travel time: {travel_time:.6f}')

    results[fn].append((travel_time, step_magnitudes))

plt.show()
```

```
/var/folders/h1/tr1q_6210b3fh7z5w19c2qvm0000gp/T/ipykernel_74397/3649079657.py:5
: RuntimeWarning: invalid value encountered in sqrt
  return 1 + np.sqrt(4 - 3 * X[:, 0] - X[:, 1])
```


GD with 5 points, travel time: 0.638590

SR1 with 5 points, travel time: 0.638593

BFGS with 5 points, travel time: 0.638676

GD with 10 points, travel time: 0.636030

SR1 with 10 points, travel time: 0.636033

BFGS with 10 points, travel time: 0.635985

GD with 20 points, travel time: 0.634745

SR1 with 20 points, travel time: 0.634730

BFGS with 20 points, travel time: 0.634731

GD with 40 points, travel time: 0.634124

SR1 with 40 points, travel time: 0.634112

BFGS with 40 points, travel time: 0.634107

GD with 80 points, travel time: 0.634520

SR1 with 80 points, travel time: 0.633839

BFGS with 80 points, travel time: 0.633844

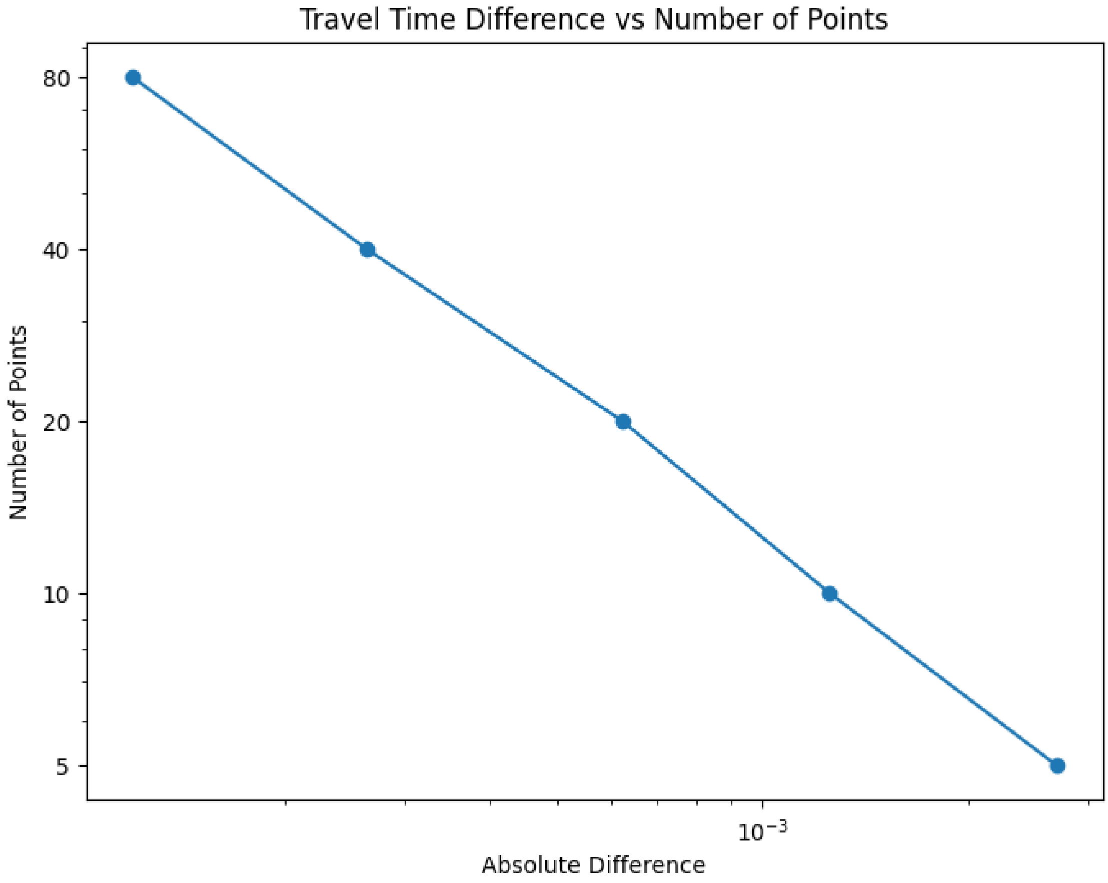
GD with 160 points, travel time: 0.634762

SR1 with 160 points, travel time: 0.634043

BFGS with 160 points, travel time: 0.633724

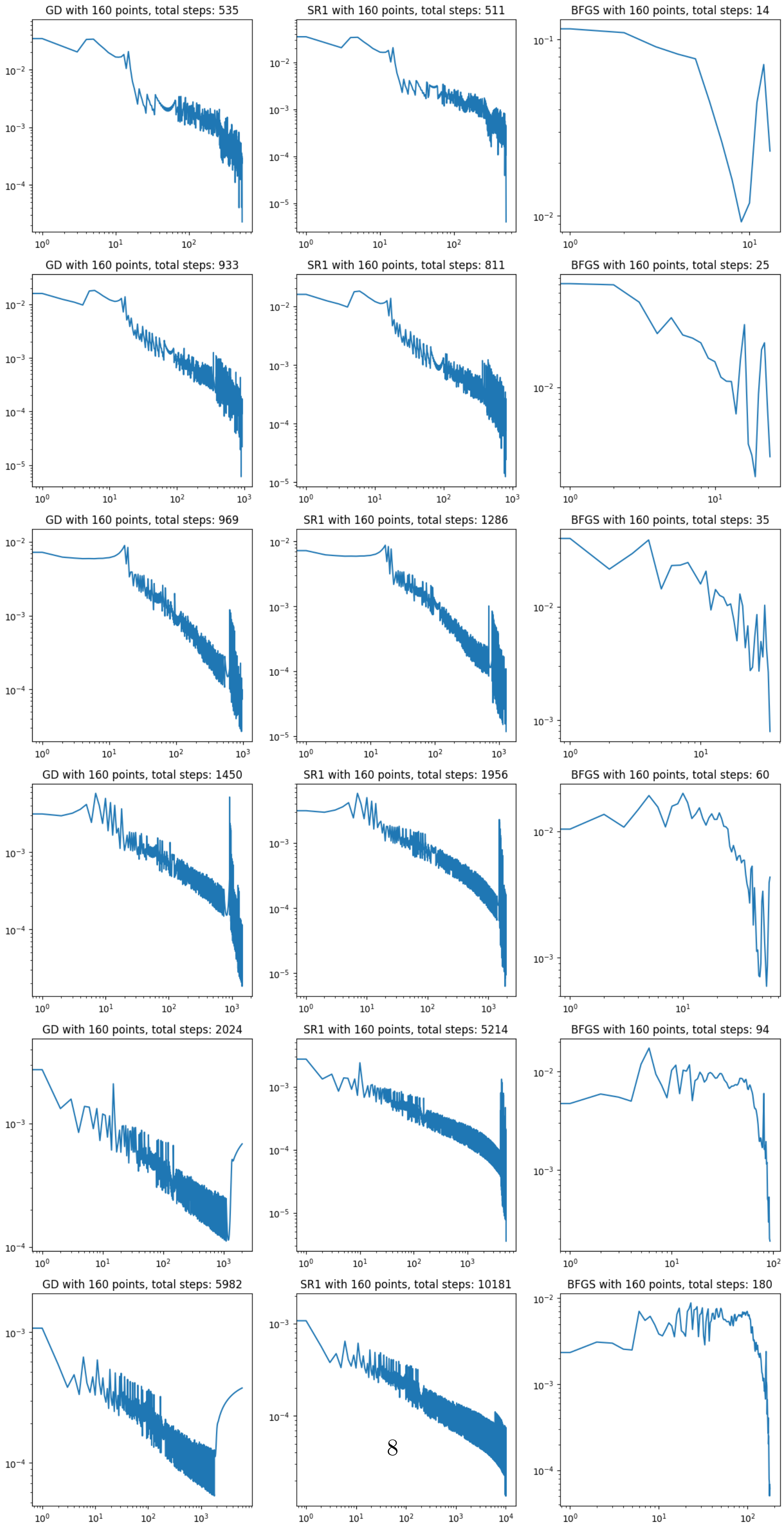
[366]:

```
abs_diff = np.abs(np.diff([result[0] for result in results[BFGS]]))
plt.figure(figsize=(8, 6))
plt.loglog(abs_diff, [5, 10, 20, 40, 80], 'o-')
# plt.xticks([1e-4, 5e-4, 1e-3, 5e-3], labels=[1e-4, 5e-4, 1e-3, 5e-3])
plt.yticks([5, 10, 20, 40, 80], labels=[5, 10, 20, 40, 80])
plt.xlabel("Absolute Difference")
plt.ylabel("Number of Points")
plt.title("Travel Time Difference vs Number of Points")
plt.show()
```



[365]:

```
fig, axs = plt.subplots(6, 3, figsize=(15, 30))

ns = [5, 10, 20, 40, 80, 160]

for (i, fn), ax in zip(itertools.product(range(6), [GD, SR1, BFGS]), axs.
  ↪ravel()):
    ax.loglog(results[fn][i][1])
    ax.set_title(
        f'{fn.__name__} with {n} points, total steps: {len(results[fn][i][1])}')

plt.show()
```


GD with 160 points, total steps: 535
10^-2
10^-3
10^-4
10^0
10^1
10^2
SR1 with 160 points, total steps: 511
10^-2
10^-3
10^-4
10^-5
10^0
10^1
10^2
BFGS with 160 points, total steps: 14
10^-1
10^-2
10^0
10^1
GD with 160 points, total steps: 933
10^-2
10^-3
10^-4
10^-5
10^0
10^1
10^2
10^3
SR1 with 160 points, total steps: 811
10^-2
10^-3
10^-4
10^-5
10^0
10^1
10^2
10^3
BFGS with 160 points, total steps: 25
10^-2
10^0
10^1
GD with 160 points, total steps: 969
10^-2
10^-3
10^-4
10^0
10^1
10^2
10^3
SR1 with 160 points, total steps: 1286
10^-2
10^-3
10^-4
10^-5
10^0
10^1
10^2
10^3
BFGS with 160 points, total steps: 35
10^-2
10^-3
10^0
10^1
GD with 160 points, total steps: 1450
10^-3
10^-4
10^0
10^1
10^2
10^3
SR1 with 160 points, total steps: 1956
10^-3
10^-4
10^-5
10^0
10^1
10^2
10^3
BFGS with 160 points, total steps: 60
10^-2
10^-3
10^0
10^1
GD with 160 points, total steps: 2024
10^-3
10^-4
10^0
10^1
10^2
10^3
SR1 with 160 points, total steps: 5214
10^-3
10^-4
10^-5
10^0
10^1
10^2
10^3
BFGS with 160 points, total steps: 94
10^-2
10^-3
10^0
10^1
10^2
GD with 160 points, total steps: 5982
10^-3
10^-4
10^0
10^1
10^2
10^3
SR1 with 160 points, total steps: 10181
10^-3
10^-4
10^0
10^1
10^2
10^3
10^4
BFGS with 160 points, total steps: 180
10^-2
10^-3
10^-4
10^0
10^1
10^2

9

[ ]:

```
```