

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

تمرین سری **چهار-پیاده‌سازی**

درس بهینه‌سازی

فرهاد دلیرانی

۹۶۱۳۱۱۲۵

dalirani@aut.ac.ir

dalirani.1373@gmail.com

فهرست

۱..... ابزارهای استفاده شده

۲..... Barrier روش

۶..... Primal-Dual Interior Point روش

ابزارهای استفاده شده

زبان برنامه نویسی: پایتون 3.6

محیط توسعه: PyCharm

سیستم عامل: Windows 10

کدهای مربوط به روش Barrier در فایل‌های زیر قرار دارد.

در این فایل شرط‌های ول قرار دارد که شرط یک آن برای Backtracking استفاده می‌شود.	Conditions.py
در این فایل تابع ϕ روش Barrier به همراه گرادیان و هسینش موجود است.	Phi.py
در این فایل تابع f_0 (تابع Quadratic که در تمرین داده شده است) و تابع $\phi + t*f_0$ قرار دارد. همچنین گرادیان و هسین این دو تابع در این فایل قرار دارند.	Function_derivatives.py
در این فایل تابعی قرار دارد که با دریافت یک تابع و مقدار x جهت کاهشی نیوتون را می‌دهد.	Search_directions.py
در این فایل تابع Backtracking برای پیدا کردن اندازه گام مناسب در جهت حرکت قرار دارد.	Step_length.py
در این فایل تابع‌هایی برای پیدا کردن کمینه یک تابع با استفاده از روش نیوتون وجود دارد که در ادامه از آن برای کمینه کردن تابع $\phi + t*f_0$ به ازای t های مختلف استفاده می‌کنیم.	Newton_descent.py
در این فایل تابع‌هایی برای رسم نمودارهای خواسته شده قرار دارد.	Plot_function.py
در این فایل تابع روش Barrier قرار دارد. با اجرای این فایل می‌توان نتایج الگوریتم را برای پنج خروجی مختلف مشاهده کرد.	Barrier_method.py

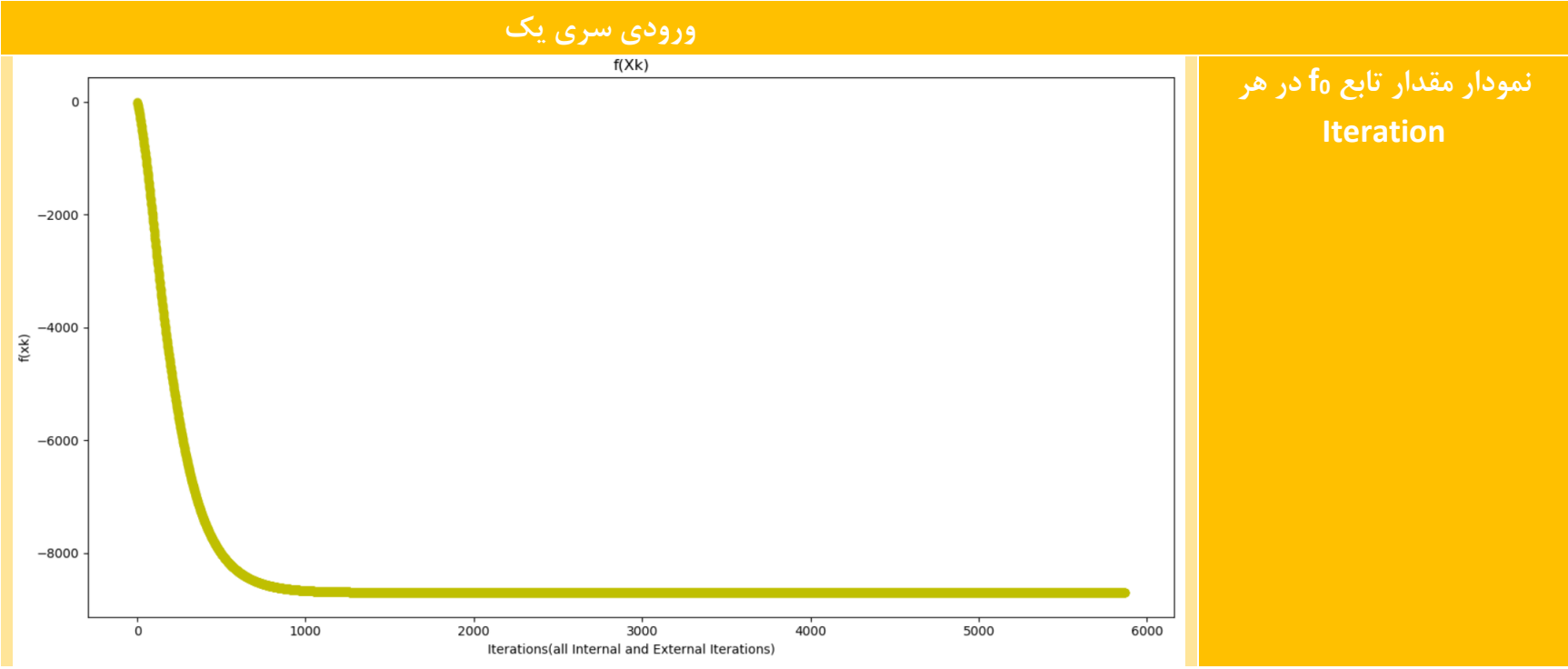
برای تولید ۵ مسئله مختلف با مقادیر مختلف A, P, q, b به صورت زیر عمل کرده‌ایم:

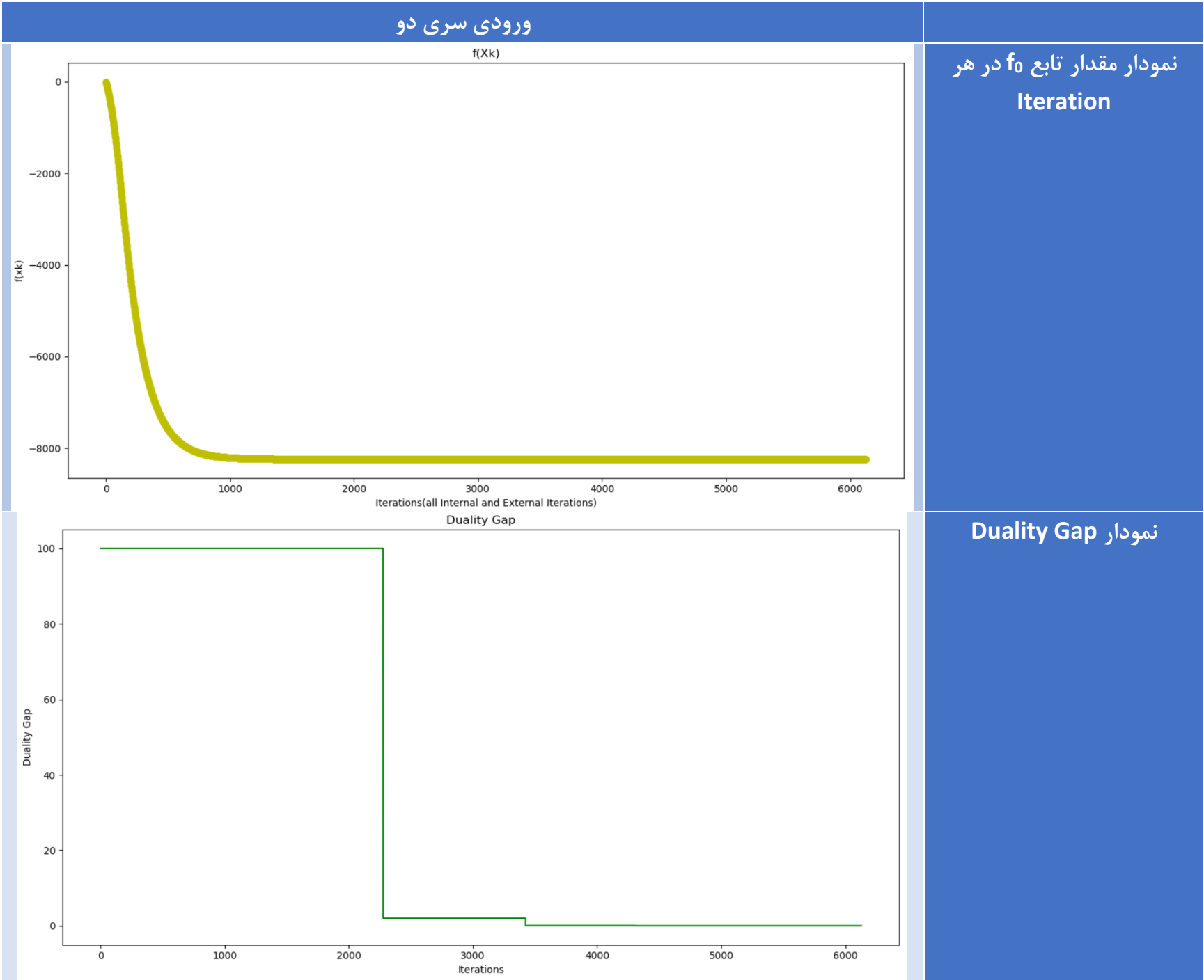
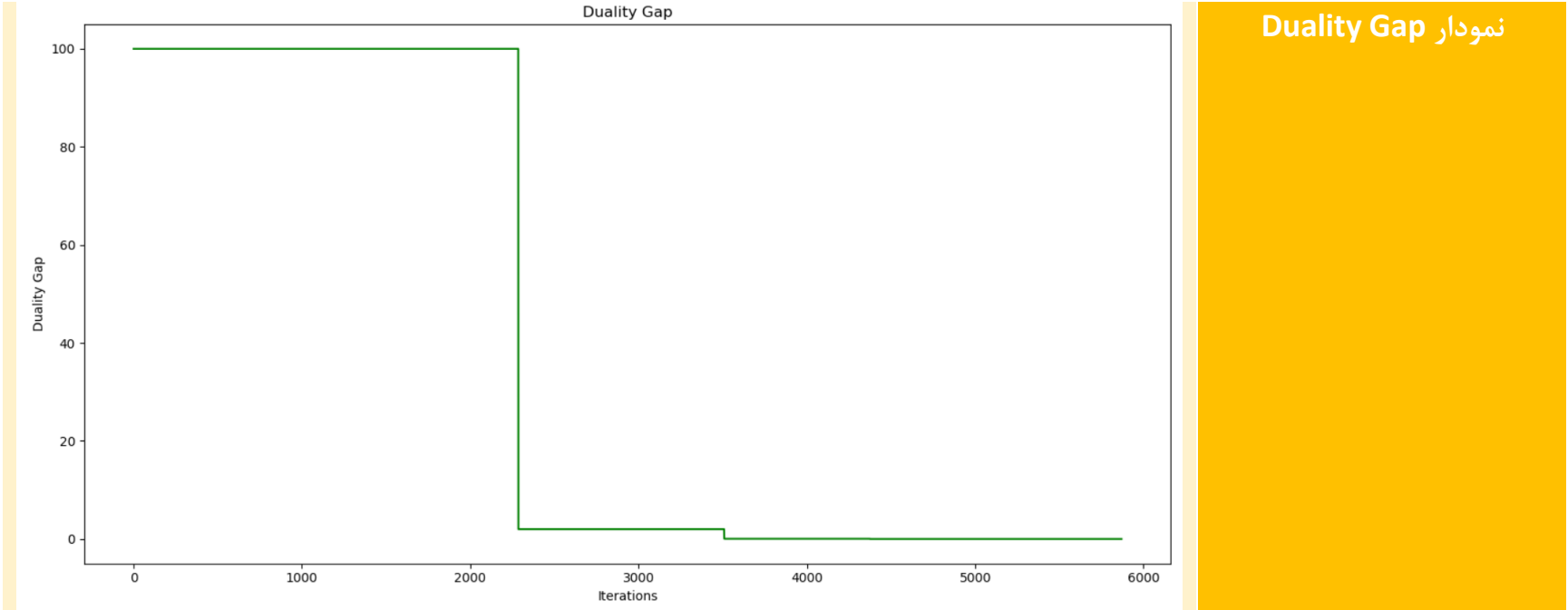
```
# for five different set of A, b, P and q solve:
# minimize      (1/2) (trans x)Px + (trans q)x
# subject to    Ax <= b
for i in range(0, 5):
    # different seed for generating different values of A, b, P and q
    np.random.seed(i)
    x0 = np.zeros((50,1))
    A = np.float32(np.random.randint(low=0, high=20, size=[100,50]))
    b = np.float32(np.random.randint(low=150, high=200, size=[100,1]))
    P = make_spd_matrix(n_dim=50)
    q = np.float32(np.random.randint(low=0, high=20, size=[50,1]))
```

مقادیر تصادفی بالا به گونه‌ای هستند که نقطه x_0 نقطه‌ای strictly feasible است. مقادیر دیگر هم تست شد که همگی توسط الگوریتم پیاده‌سازی شده حل شدند.

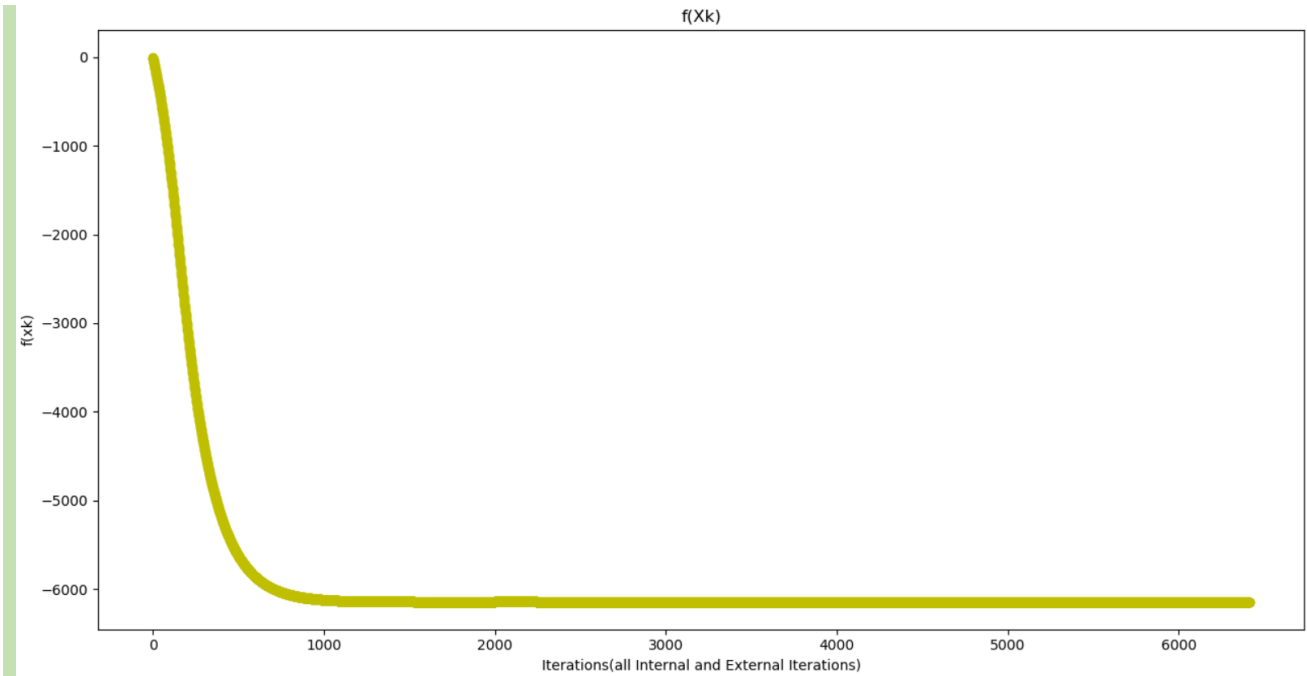
T_0 مقداری برابر با یک دارد. مقدار μ برابر با 50 قرار داده شده است و مقدار ϵ برابر با 10^{-6} است.

در زیر خروجی‌های الگوریتم برای ۵ دسته متفاوت از ورودی‌ها را مشاهده می‌کنید:

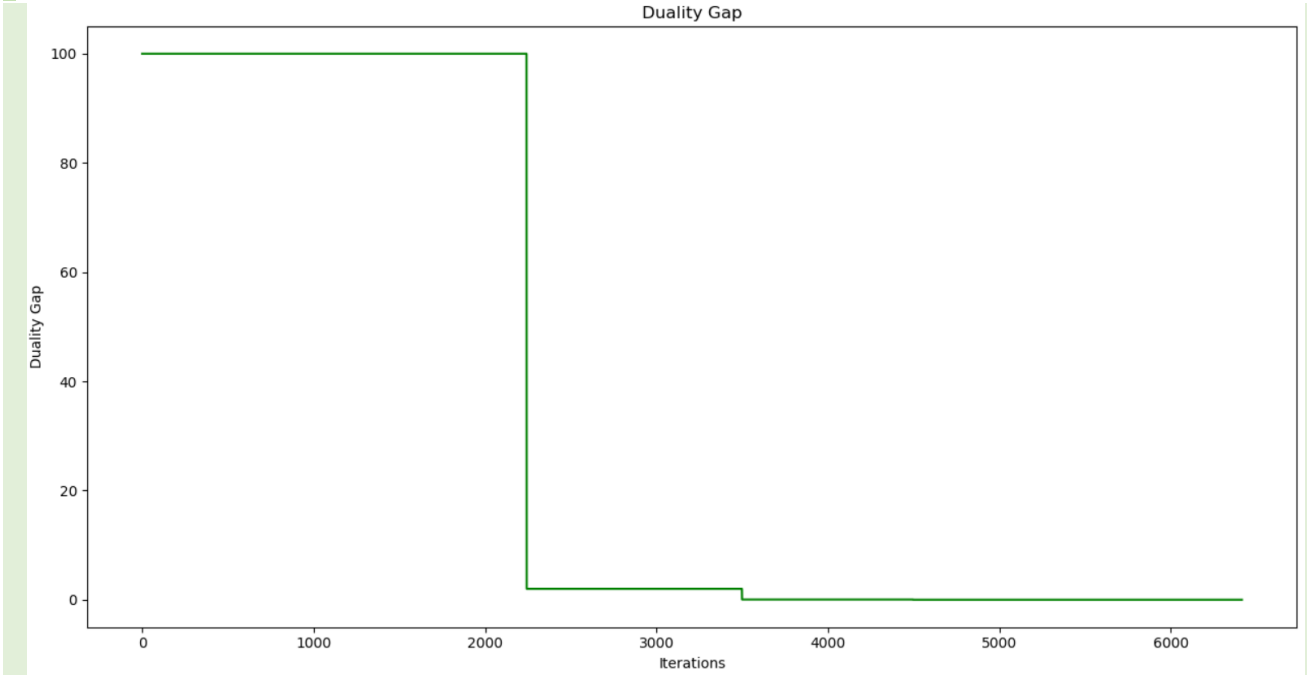




ورودی سری سه	
--------------	--

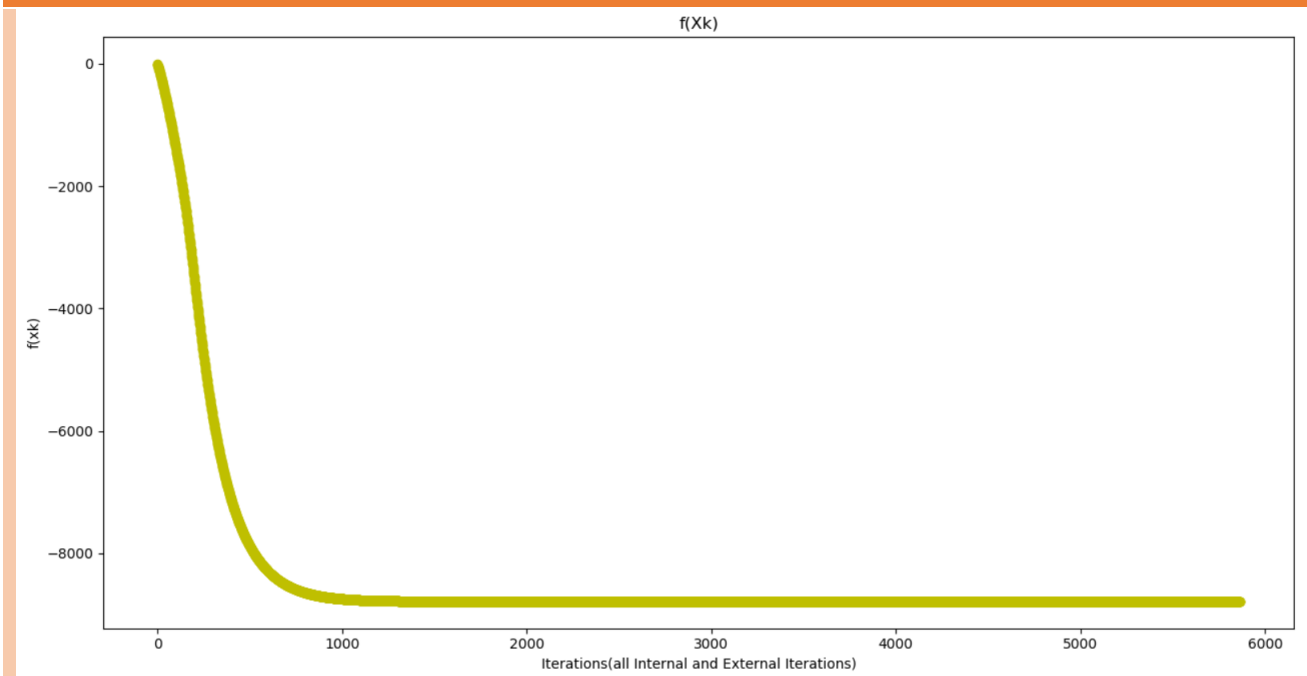


نمودار مقدار تابع f_0 در هر Iteration

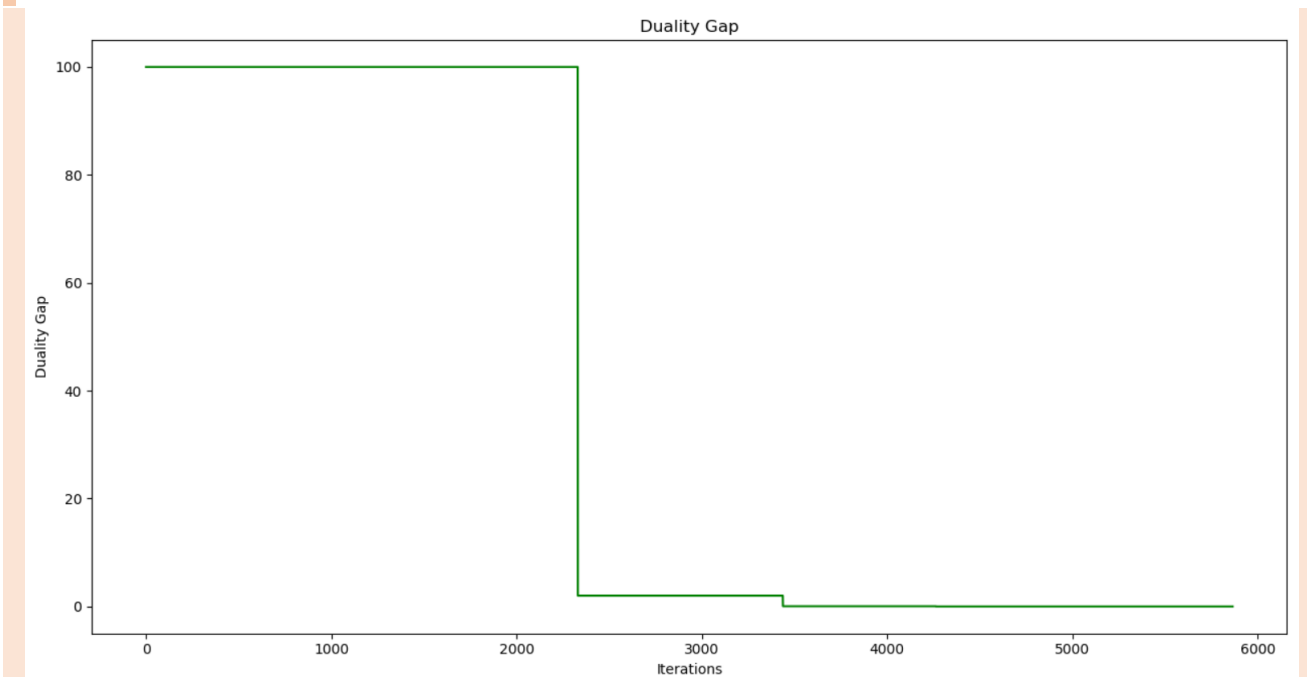


نمودار Duality Gap

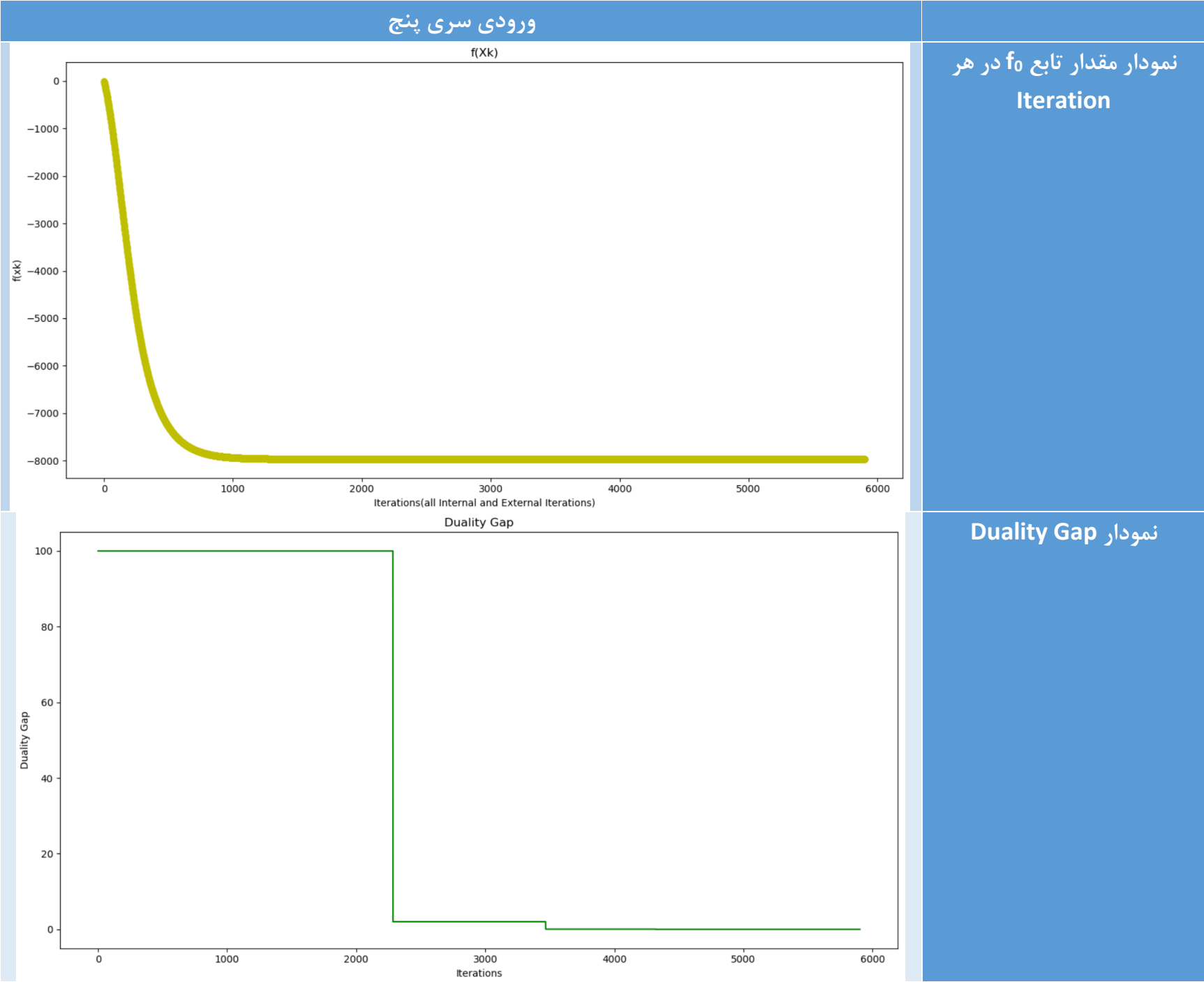
ورودی سری چهار



نمودار مقدار تابع f_0 در هر Iteration



نمودار Duality Gap



کدهای مربوط به روش Primal-Dual Interior Point در فایل‌های زیر قرار دارد.

در این فایل تابع‌های مختلف مورد نیاز نوشته شده است. تابع f_0 (تابع Quadratic که در تمرین داده شده است) و تابع گرادیان و هسین آن قرار دارد. در این فایل تابع $f=(f_1,f_2,...,f_m)$ و مشتقش قرار دارد. همچنین تابع‌های r_{cent} ، r_{dual} و surrogate duality gap هم در این فایل موجوداند.	Primal_dual_functions_drivatives.py
در این فایل تابع‌هایی برای رسم نمودارهای خواسته شده قرار دارد.	Plot_function.py
در این فایل تابع‌هایی برای پیاده‌سازی روش Primal-Dual Interior Point قرار دارد. تابع compute_primal_dual_direction با حل دستگاه $\Delta y = -Dr_t(y)^{-1}r_t(y)$ جهت حرکت را مشخص می‌کند. تابع backtracking اندازه گام برای Δy را مشخص می‌کند. از روش line search موجود در صفحه‌ی ۶۱۲ کتاب بهینه‌سازی بوید استفاده شده است. تابع primal_dual_interior_point_method پیاده‌سازی روش Primal Dual Interior Point است. با اجرای این فایل نتایج را برای ۵ دسته ورودی متفاوت می‌توان دید.	Primal_dual_interior_point.py

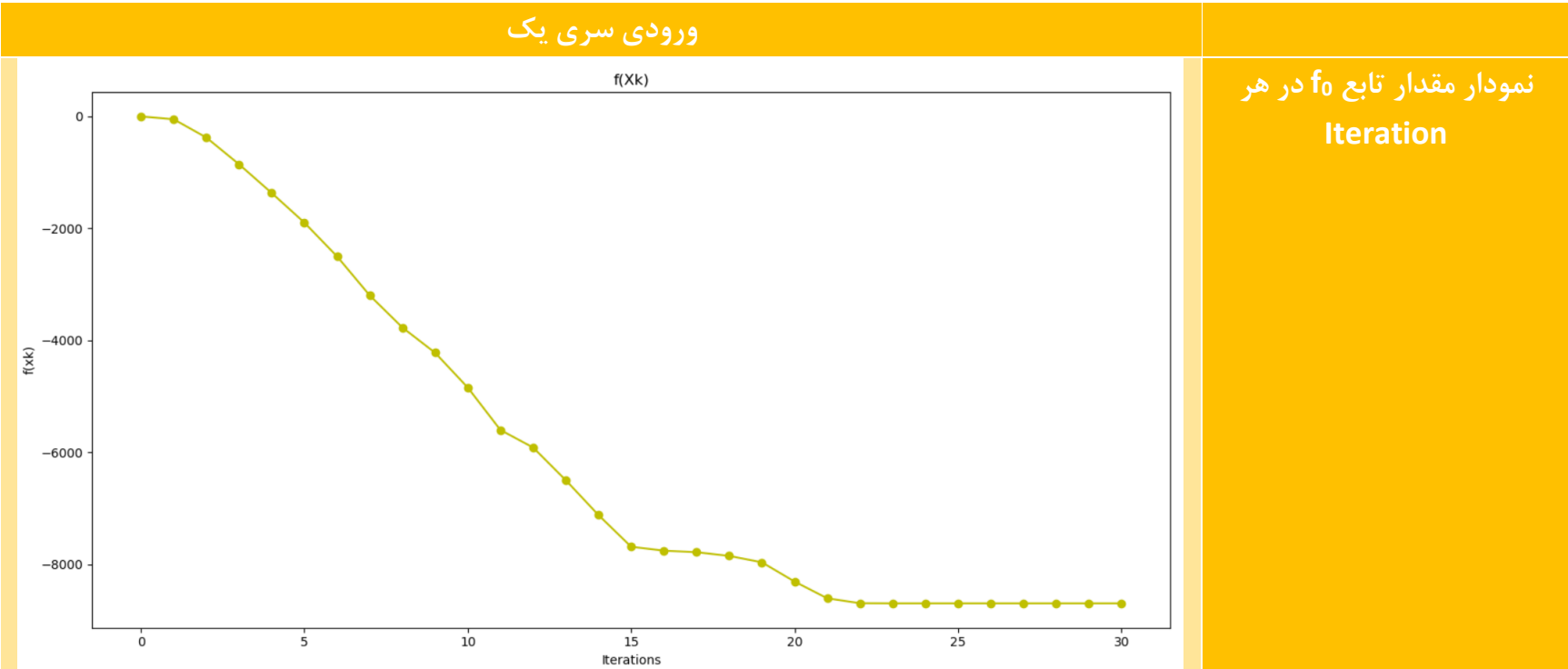
برای تولید ۵ مسئله مختلف با مقادیر مختلف A, P, q, b به صورت زیر عمل کرده‌ایم:

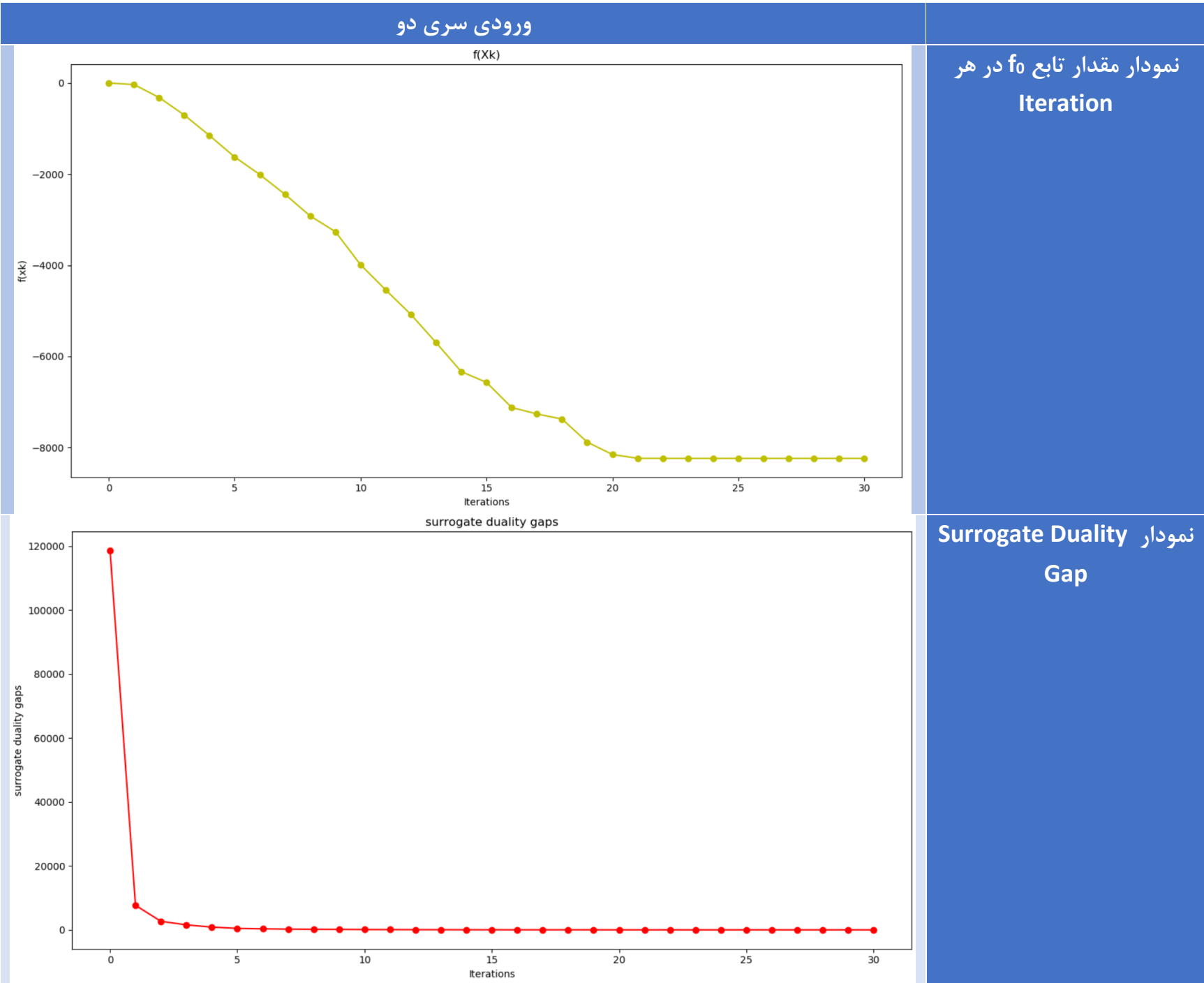
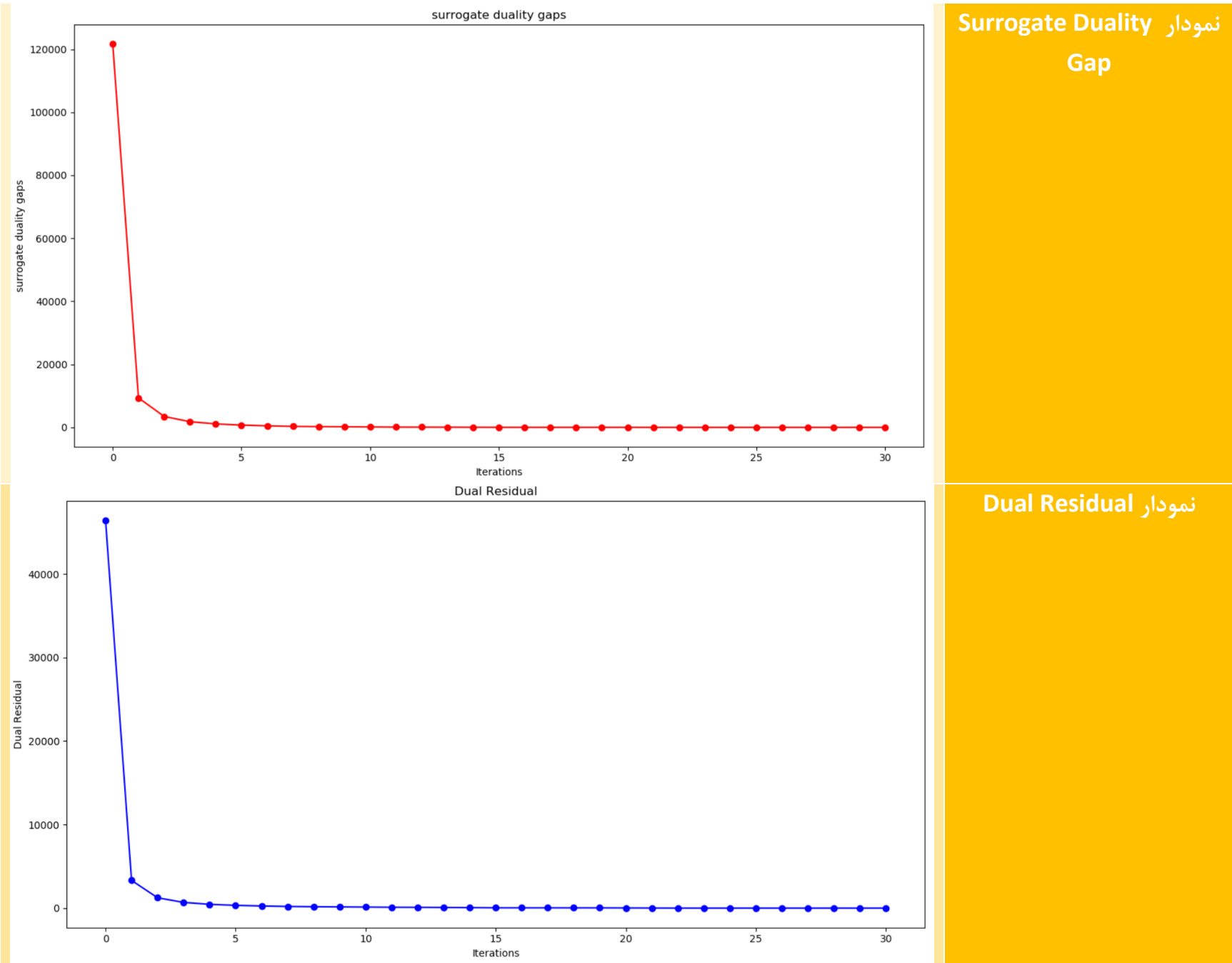
```
# for five different set of A, b, P and q solve:
# minimize      (1/2) (trans x)Px + (trans q)x
# subject to    Ax <= b
for i in range(0, 5):
    # different seed for generating different values of A, b, P and q
    np.random.seed(i)
    x0 = np.zeros((50,1))
    A = np.float32(np.random.randint(low=0, high=20, size=[100,50]))
    b = np.float32(np.random.randint(low=150, high=200, size=[100,1]))
    P = make_spd_matrix(n_dim=50)
    q = np.float32(np.random.randint(low=0, high=20, size=[50,1]))
```

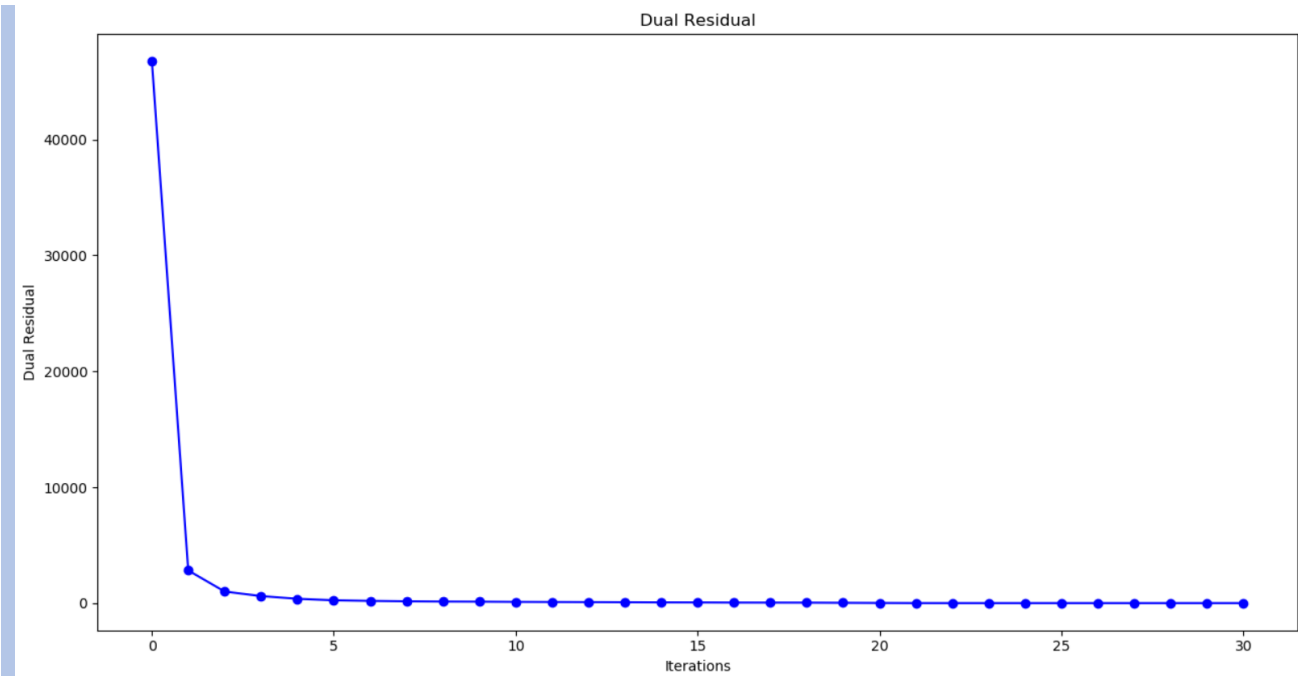
مقادیر تصادفی بالا به گونه‌ای هستند که نقطه x_0 نقطه‌ای strictly feasible است. مقادیر دیگر هم تست شد که همگی توسط الگوریتم پیاده‌سازی شده حل شدند.

مقدار μ برابر با 10 قرار داده شده است و مقدار ϵ برابر با 1×10^{-6} و مقدار ϵ_{feas} برابر با 1×10^{-6} است.

در زیر خروجی‌های الگوریتم برای ۵ دسته متفاوت از ورودی‌ها را مشاهده می‌کنید:

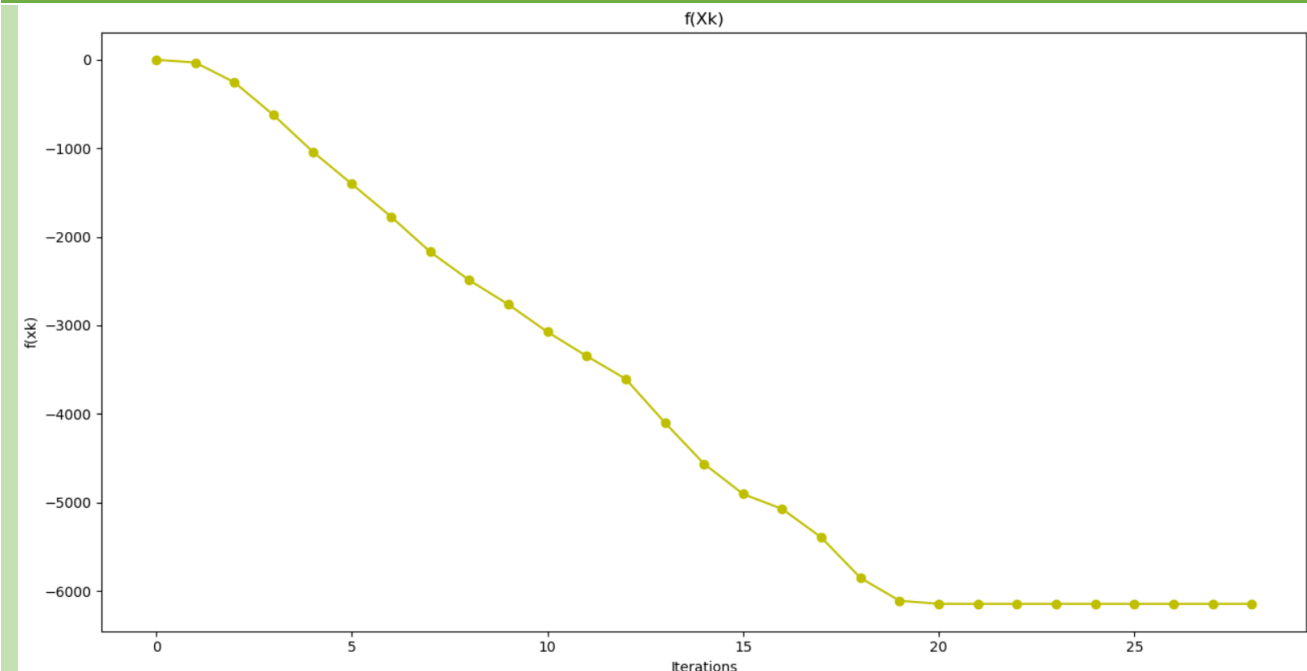




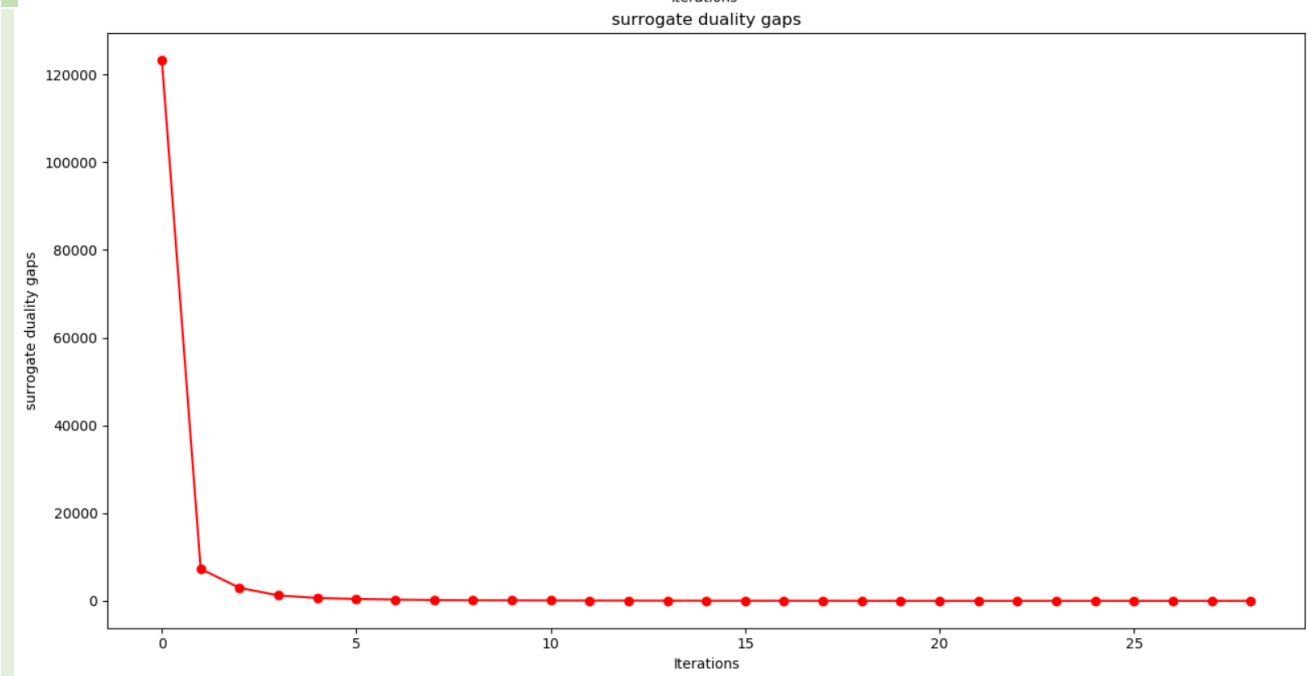


نمودار Dual Residual

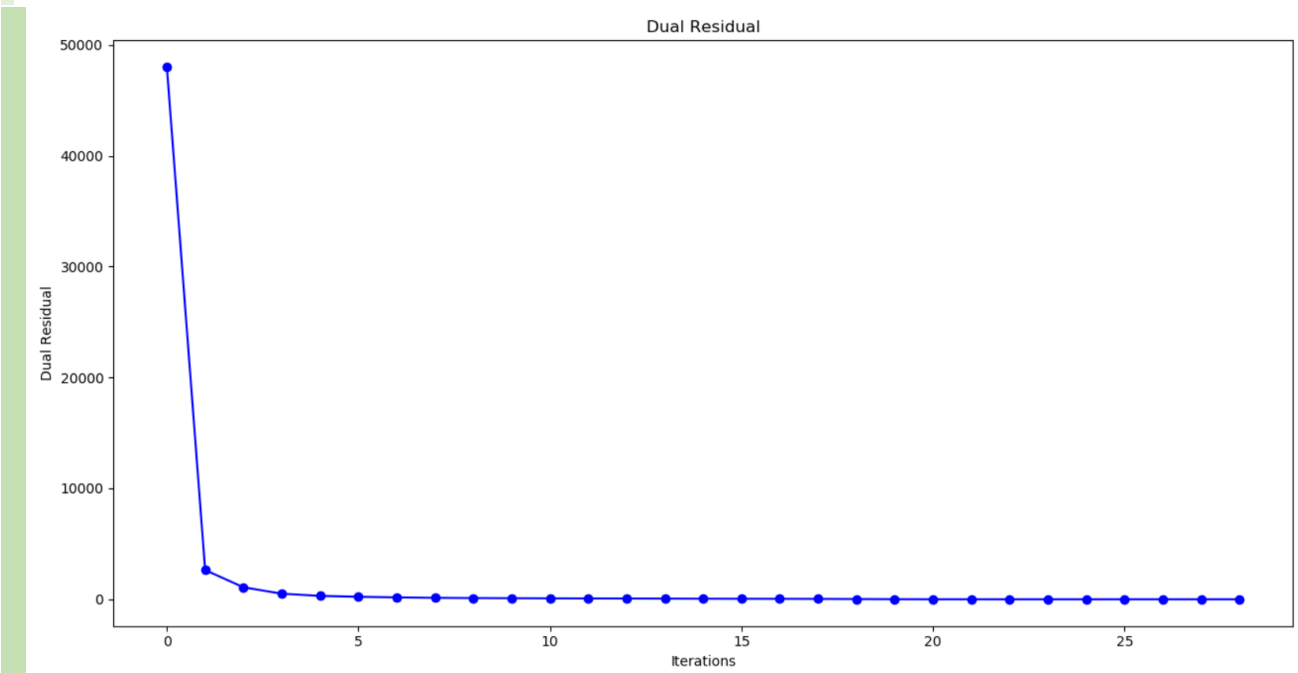
ورودی سری سه



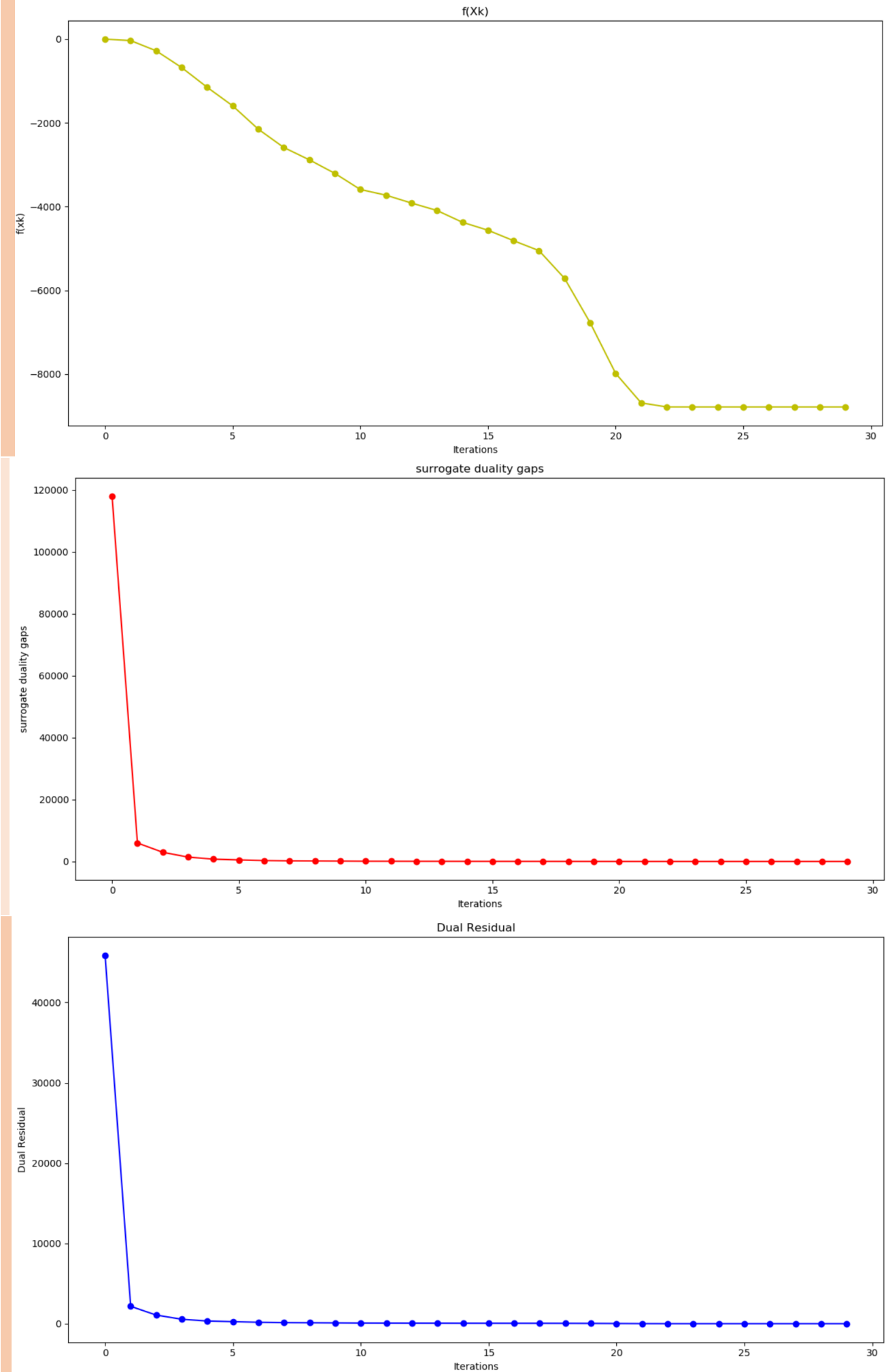
نمودار مقدار تابع f_0 در هر Iteration

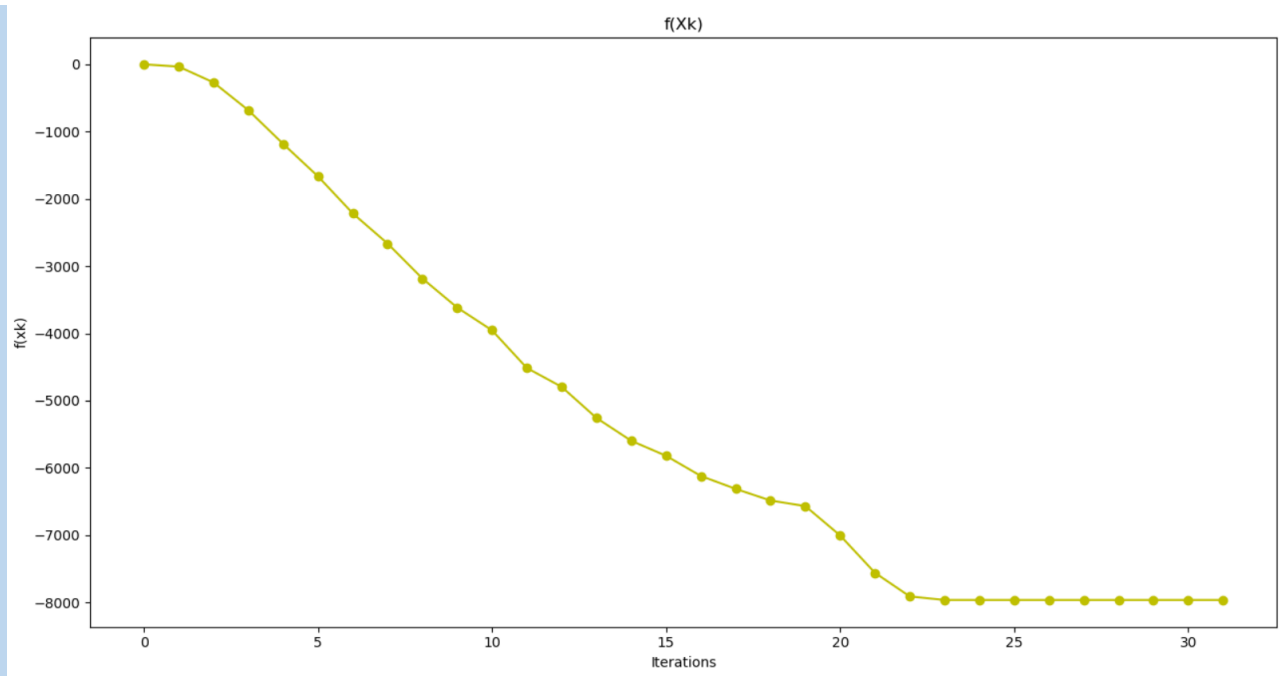


نمودار Surrogate Duality Gap

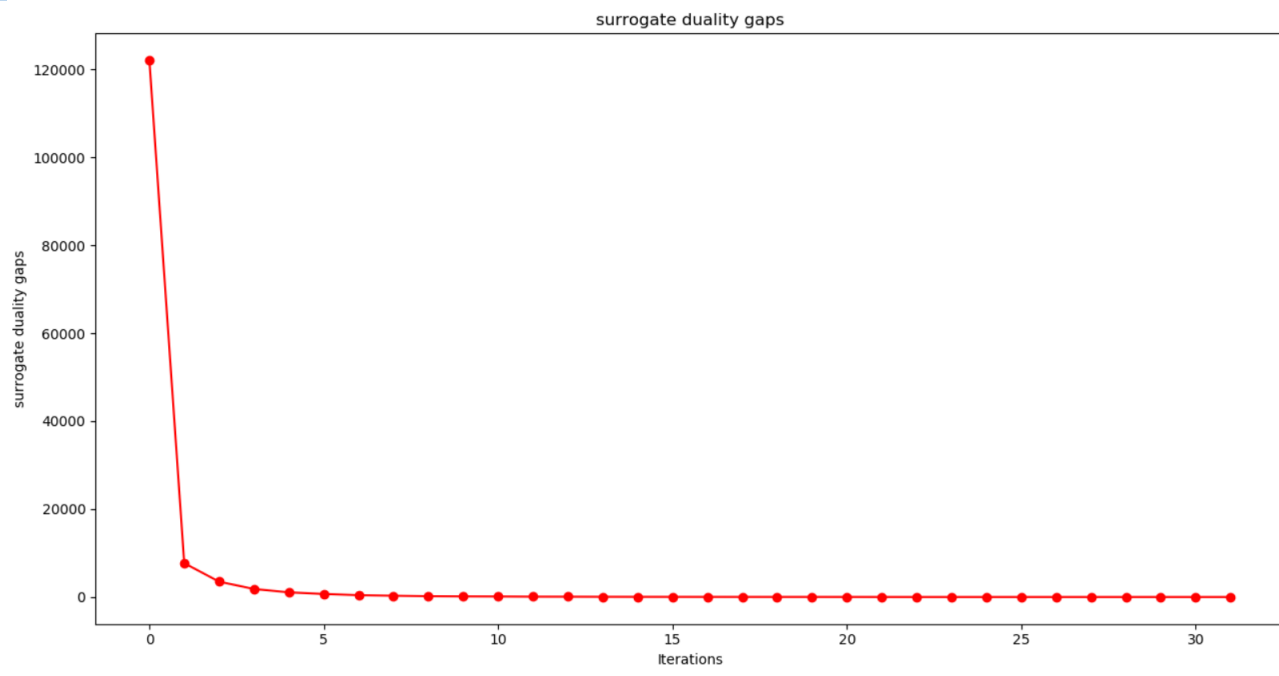


نمودار Dual Residual

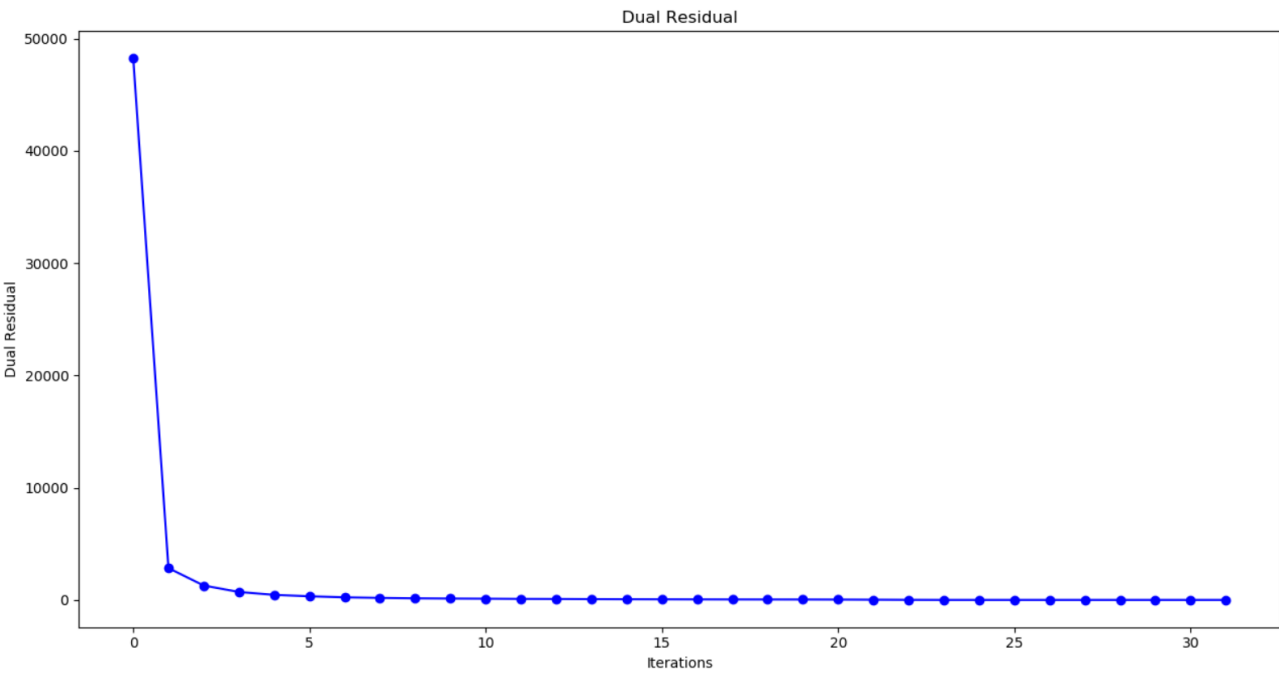
ورودی سری چهار		
	<p>نمودار مقدار تابع f_0 در هر Iteration</p>	
	<p>نمودار Surrogate Duality Gap</p>	
	<p>نمودار Dual Residual</p>	
ورودی سری پنج		



نمودار مقدار تابع f_0 در هر Iteration



نمودار Surrogate Duality Gap



نمودار Dual Residual