# Fungsi Tujuan Tanpa Kendala (Unconstrained Optimization)

## Pendahuluan

Dalam **optimasi nonlinier**, terdapat dua kategori besar:

1. **Optimasi tanpa kendala** (*Unconstrained Optimization*):
   * Tidak ada batasan/constraint pada variabel keputusan.
   * Contoh: meminimalkan dengan .
2. **Optimasi dengan kendala** (*Constrained Optimization*):
   * Ada syarat tambahan seperti atau .
   * Contoh: meminimalkan dengan syarat .

Fungsi tujuan tanpa kendala adalah bentuk paling dasar, menjadi dasar untuk memahami kasus lebih kompleks dengan kendala.

## Definisi Formal

Masalah umum tanpa kendala ditulis:

dengan:

* → fungsi tujuan (objective function).
* → vektor variabel keputusan berukuran .

Tidak ada kendala tambahan (baik kesamaan maupun ketaksamaan).

## Konsep Titik Optimum:

### a. Titik minimum global

adalah minimum global jika:

### b. Titik minimum lokal

adalah minimum lokal jika:

Dalam praktik, yang sering ditemukan adalah minimum lokal (karena fungsi nonlinier bisa punya banyak titik ekstrem).

# Contoh Fungsi tujuan tanpa kendala

**1. Fungsi yang diberikan:**

### Analisis:

* Turunan pertama:
* Titik stasioner: .
* Turunan kedua:

Hasil: Minimum di , nilai minimum .

# 2. Fungsi Polinomial Kubik

### Analisis:

* Turunan pertama:
* Titik stasioner: dan .
* Turunan kedua:
  + → maksimum lokal di .
  + → minimum lokal di .

Hasil: Maksimum di , Minimum di .

# 3. Fungsi Eksponensial

### Analisis:

* Turunan pertama:
* Tidak ada titik stasioner → fungsi selalu naik.

Hasil: Tidak ada minimum/maksimum lokal (fungsi monoton naik).

# 4. Fungsi Trigonometri

### Analisis:

* Turunan pertama:
* Titik stasioner: .
* Turunan kedua:
  + → maksimum lokal.
  + → minimum lokal.

Hasil: Fungsi periodik dengan maksimum di dan minimum di .

# 5. Fungsi Dua Variabel (Contoh Nonlinear Klasik)

### Analisis:

* Gradien:
* Titik stasioner: .
* Hessian:
* Semua nilai eigen positif → definit positif.

## 

## Kondisi Optimalitas

### a. Kondisi Orde Pertama (First-Order Necessary Condition)

Jika terdiferensial di , maka gradien di titik optimal bernilai nol:

Artinya: tidak ada arah penurunan lagi pada titik tersebut.

### b. Kondisi Orde Kedua (Second-Order Condition)

Gunakan Hessian matrix .

* Jika positif definit → adalah minimum lokal.
* Jika negatif definit → adalah maksimum lokal.
* Jika indefinit → adalah saddle point.

## Apa itu Saddle Point dalam Optimasi Nonlinear?

Dalam optimasi nonlinear, kita sering menganalisis fungsi tujuan untuk mencari titik-titik ekstremum (minimum atau maksimum lokal). Namun, tidak semua titik stasioner (titik di mana gradien = 0) merupakan minimum atau maksimum. Ada juga kemungkinan munculnya saddle point.

### Definisi Saddle Point

Saddle Point adalah titik stasioner pada suatu fungsi di mana:

* Fungsi tersebut tidak mencapai minimum maupun maksimum lokal.
* Fungsi naik di satu arah, tetapi turun di arah lain.

Dengan kata lain, di sekitar titik itu, fungsi berbentuk seperti pelana kuda (saddle), ada bagian yang melengkung ke atas, ada bagian yang melengkung ke bawah.

### Contoh Geometris

Misalkan fungsi dua variabel:

* Di titik , gradien → ini titik stasioner.
* Tetapi:
  + Jika kita bergerak sepanjang sumbu , → berbentuk minimum.
  + Jika kita bergerak sepanjang sumbu , → berbentuk maksimum.
* Jadi, titik bukan minimum atau maksimum, melainkan saddle point.

### Karakterisasi Saddle Point (dengan Hessian)

Untuk fungsi dua variabel , Hessian adalah:

Kriteria:

1. Jika Hessian definit positif → titik stasioner adalah minimum lokal.
2. Jika Hessian definit negatif → titik stasioner adalah maksimum lokal.
3. Jika Hessian indefinit (ada nilai eigen positif dan negatif) → titik stasioner adalah saddle point.

## 

## Metode Penyelesaian

Karena biasanya tidak ada solusi tertutup untuk fungsi nonlinier kompleks, digunakan metode numerik:

### 🔹 1. Gradient Descent (Steepest Descent)

Iterasi:

* = *step size* (learning rate).
* Cocok untuk fungsi halus, sederhana, tetapi bisa lambat jika kondisi buruk.

### 🔹 2. Newton’s Method

Menggunakan informasi Hessian:

* Konvergensi sangat cepat (kuadratik) bila dekat solusi.
* Biaya mahal untuk dimensi tinggi (perlu invers Hessian).

### 🔹 3. Quasi-Newton (misal: BFGS, L-BFGS)

* Aproksimasi invers Hessian.
* Lebih efisien untuk masalah dimensi besar.

### 🔹 4. Conjugate Gradient

* Khusus untuk fungsi kuadratik dengan matriks simetris positif definit.
* Banyak dipakai dalam optimasi komputasi besar.

**Contoh-contoh fungsi kendala tanpa tujuan dengan codingan program python:**

🔹 1. Fungsi Kuadrat

Coding program untuk fungsi tersebut:

import numpy as np

import matplotlib.pyplot as plt

# Fungsi

x = np.linspace(-6, 2, 400)

f = x\*\*2 + 4\*x + 5

# Titik minimum

x\_min = -2

y\_min = f.min()

# Plot

plt.figure(figsize=(6,4))

plt.plot(x, f, label="f(x)=x²+4x+5")

plt.scatter(x\_min, x\_min\*\*2 + 4\*x\_min + 5, color='red', label="Minimum (-2,1)")

plt.title("Fungsi Kuadrat")

plt.xlabel("x")

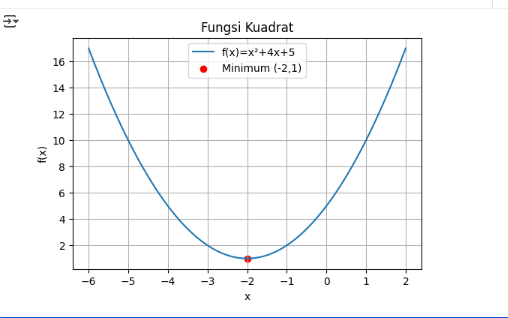
plt.ylabel("f(x)")

plt.legend()

plt.grid(True)

plt.show()

hasil coding program diatas:



# 🔹 2. Fungsi Polinomial Kubik

Coding program untuk fungsi tersebut:

import numpy as np

import matplotlib.pyplot as plt

# Fungsi Kubik

x = np.linspace(-1, 3, 400)

f = x\*\*3 - 3\*x\*\*2 + 2

# Titik stasioner

x\_stat = [0, 2]

y\_stat = [x\*\*3 - 3\*x\*\*2 + 2 for x in x\_stat]

# Plot

plt.figure(figsize=(6,4))

plt.plot(x, f, label="f(x)=x³-3x²+2")

plt.scatter(x\_stat, y\_stat, color='red', label="Titik stasioner")

plt.title("Fungsi Kubik")

plt.xlabel("x")

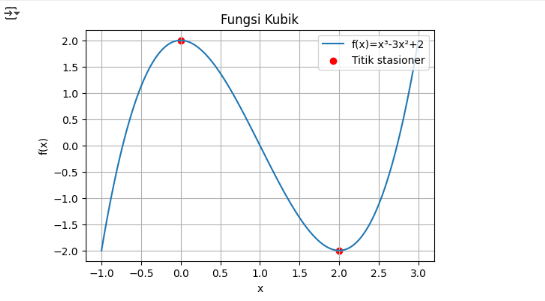
plt.ylabel("f(x)")

plt.legend()

plt.grid(True)

plt.show()

hasil coding program diatas:



# 🔹 3. Fungsi Eksponensial

Coding program untuk fungsi tersebut:

import numpy as np

import matplotlib.pyplot as plt

# Fungsi Eksponensial

x = np.linspace(-2, 2, 400)

f = np.exp(x)

# Plot

plt.figure(figsize=(6,4))

plt.plot(x, f, label="f(x)=e^x")

plt.title("Fungsi Eksponensial (Monoton Naik)")

plt.xlabel("x")

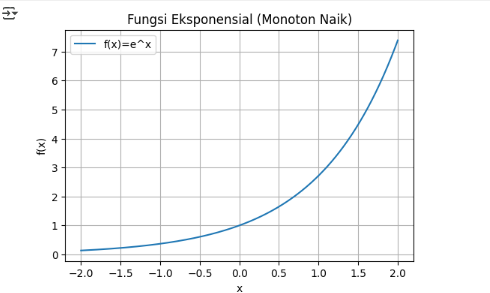
plt.ylabel("f(x)")

plt.legend()

plt.grid(True)

plt.show()

hasil coding program diatas:



# 🔹 4. Fungsi Trigonometri

Coding program untuk fungsi tersebut:

import numpy as np

import matplotlib.pyplot as plt

# Fungsi Trigonometri

x = np.linspace(-2\*np.pi, 2\*np.pi, 400)

f = np.sin(x)

# Titik ekstrem

x\_max = np.pi/2

x\_min = 3\*np.pi/2

y\_max = np.sin(x\_max)

y\_min = np.sin(x\_min)

# Plot

plt.figure(figsize=(6,4))

plt.plot(x, f, label="f(x)=sin(x)")

plt.scatter([x\_max, x\_min], [y\_max, y\_min], color='red', label="Ekstremum")

plt.title("Fungsi Trigonometri")

plt.xlabel("x")

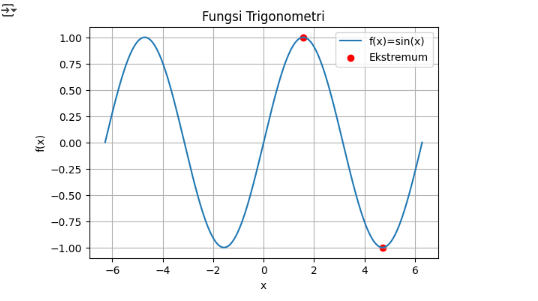
plt.ylabel("f(x)")

plt.legend()

plt.grid(True)

plt.show()

hasil coding program diatas:



# 🔹 5. Fungsi Dua Variabel (Contoh Nonlinear Klasik)

Coding program untuk fungsi tersebut:

import numpy as np

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

# Fungsi Dua Variabel

x = np.linspace(-2, 2, 100)

y = np.linspace(-2, 2, 100)

X, Y = np.meshgrid(x, y)

Z = X\*\*2 + Y\*\*2

# Plot 3D

fig = plt.figure(figsize=(8,6))

ax = fig.add\_subplot(111, projection='3d')

ax.plot\_surface(X, Y, Z, cmap="viridis", alpha=0.8)

ax.scatter(0, 0, 0, color="red", s=50, label="Minimum (0,0)")

ax.set\_title("Fungsi Dua Variabel: f(x,y)=x²+y²")

ax.set\_xlabel("x")

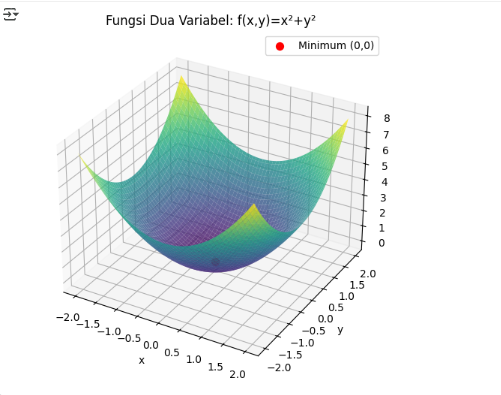
ax.set\_ylabel("y")

ax.set\_zlabel("f(x,y)")

ax.legend()

plt.show()

hasil coding program diatas:



# Penjelasan dari contoh-contoh fungsi kendala tanpa tujuan

# 🔹 1. Fungsi Kuadrat

Grafik: parabola terbuka ke atas.

* Kurva parabola menunjukkan fungsi yang selalu naik ke arah kiri dan kanan.
* Titik terendah (minimum) ada di dengan nilai .
* Artinya, solusi optimal (minimum) untuk fungsi ini adalah pada titik .

# 🔹 2. Fungsi Kubik

Grafik: berbentuk huruf S.

* Fungsi memiliki dua titik stasioner:
  + → menghasilkan puncak (maksimum lokal).
  + → menghasilkan lembah (minimum lokal).
* Fungsi ini menunjukkan perilaku khas polinomial kubik: ada daerah naik–turun–naik.

# 🔹 3. Fungsi Eksponensial

Grafik: kurva eksponensial yang selalu naik.

* Fungsi ini tidak memiliki titik stasioner karena turunan selalu positif.
* Artinya, fungsi selalu meningkat tanpa ada maksimum atau minimum.
* Dalam konteks optimasi, fungsi ini tidak punya solusi ekstremum kecuali diberi batasan (kendala).

# 🔹 4. Fungsi Trigonometri

Grafik: bergelombang (periodik) antara dan .

* Fungsi memiliki pola naik–turun yang berulang.
* Maksimum terjadi pada dengan nilai 1.
* Minimum terjadi pada dengan nilai -1.
* Fungsi ini menggambarkan fenomena siklus/periodik, sangat relevan untuk data musiman atau sinyal.

# 🔹 5. Fungsi Dua Variabel

Grafik: permukaan berbentuk mangkuk (paraboloid).

* Semua nilai positif kecuali di titik pusat .
* Titik adalah minimum global karena semua arah menuju ke nilai yang lebih tinggi.
* Fungsi ini sering dijadikan contoh klasik dalam optimasi multivariat.

Kesimpulan Umum:

* Fungsi kuadrat dan paraboloid → memiliki minimum global yang jelas.
* Fungsi kubik → memiliki maksimum dan minimum lokal.
* Fungsi eksponensial → tidak ada titik ekstrem, selalu naik.
* Fungsi sinus → memiliki maksimum dan minimum periodik.

# Fungsi Tujuan dengan Kendala

## 1. Gambaran umum

Masalah optimasi berkendala berarti kita ingin mencari nilai yang meminimalkan (atau memaksimalkan) suatu fungsi tujuan , tetapi harus memenuhi satu atau lebih syarat/kendala. Bentuk umum (minimisasi):

* = kendala *kesetaraan* (equality).
* = kendala *ketaksamaan* (inequality).
* Himpunan semua yang memenuhi semua kendala disebut feasible set.

Visualisasi mudah: bayangkan kontur fungsi (garis level). Feasible set memotong permukaan itu; optimum terletak di tempat kontur paling “rendah” yang masih berada di area feasible — mungkin di interior, atau di boundary (biasa terjadi).

## 2. Geometri: interior vs boundary vs active set

* Interior point: titik feasible yang semua kendala ketaksamaan tidak “menempel” (semua ). Di interior berlaku kondisi gradien = 0 seperti tanpa kendala.
* Boundary / aktif: kendala yang menempel di solusi (untuk : ) disebut active constraints. Hanya gradien luruh ke kombinasi gradien kendala yang bisa membatalkan gradien fungsi tujuan.
* Active set: himpunan kendala yang aktif di titik tertentu — penting secara algoritmik (active-set methods).

## 3. Lagrange multipliers — kasus kesetaraan (paling dasar)

Untuk kendala kesetaraan (anggap satu kendala dulu), kita bentuk Lagrangian:

Kondisi stasioner (first order necessary):

Interprestasi: gradien di titik optimum harus bisa ditulis sebagai kombinasi linear gradien kendala — artinya tidak ada arah di feasible set yang menurunkan .

### Contoh langkah-langkah (sederhana)

Minimalkan s.t. .

1. Lagrangian: .
2. Stasioner: , . → .
3. Constraint: .
4. Jadi solusi . Lagrange multiplier .

## 4. KKT (Karush–Kuhn–Tucker) — kendala umum (eq + ineq)

KKT adalah ekstensi Lagrange untuk kombinasi equality & inequality. Bentuk Lagrangian lengkap:

dengan syarat .

KKT conditions (yang harus dipenuhi pada solusi bila constraint qualifications terpenuhi):

1. **Primal feasibility**: , .
2. **Dual feasibility**: untuk semua .
3. **Stationarity**:
4. **Complementary slackness**:

* Artinya: tiap kendala ketaksamaan yang tidak aktif () harus punya ; bila kendala aktif (), multiplier boleh >0.

Catatan: Untuk masalah konveks (f konveks, g\_i konveks, h\_j linear), KKT adalah kondisi *cukup* juga (tidak hanya perlu): titik yang memenuhi KKT adalah optimum global.

### Intuisi multiplier (shadow price)

Lagrange multipliers memberi tahu “berapa banyak nilai fungsi tujuan meningkat jika kendala dilonggarkan sedikit” — interpretasi ekonomis: *harga bayangan*.

## 5. Contoh KKT sederhana (1D inequality)

Minimalkan s.t. . Tuliskan .

Lagrangian: .

KKT:

* Primal: .
* Dual: .
* Stationarity:
* Complementary slackness: .

Analisis:

* Jika kendala inactive (), maka → → tidak mungkin.
* Jadi kendala aktif: . Lalu . Solusi , .

## 6. Constraint qualifications (mengapa perlu?)

KKT hanya *perlu* bila beberapa kondisi regularitas terpenuhi — itulah *constraint qualifications (CQ)*. CQ paling umum:

* LICQ (Linear Independence Constraint Qualification): gradien semua kendala aktif di titik solusi adalah linier-independen.
* MFCQ (Mangasarian–Fromovitz): lebih lemah, sering dipakai.
* Slater’s condition (khusus konveks): ada titik yang memenuhi semua ketaksamaan secara *ketat* → menjamin strong duality.

Jika CQ dilanggar, KKT mungkin gagal (multipliers tidak ada, atau kondisi tak berlaku).

## 7. Dualitas (secara intuitif)

* Bentuk dual melalui fungsi dual .
* Weak duality: dual objective ≤ primal objective for any feasible pair → dual memberi *lower bound* bagi masalah minimisasi.
* Strong duality: optimal primal = optimal dual (terjadi pada banyak masalah konveks jika Slater terpenuhi). Strong duality memungkinkan menyelesaikan masalah melalui dual (kadang lebih mudah).

## 8. Metode numerik populer (intuitif & kapan dipakai)

1. Active-set methods
   * Kerja: tebak kumpulan kendala aktif; selesaikan subproblem equality-constrained; perbarui aktifitas. Baik untuk QP dan masalah ukuran kecil–sedang, berguna bila solusi sparse active set.
2. Sequential Quadratic Programming (SQP)
   * Setiap iterasi buat subproblem QP (aproksimasi Lagrangian + linearization constraints). Mirip Newton untuk masalah berkendala. Sangat efektif untuk masalah smooth nonkonveks berdimensi sedang.
3. Interior-point methods (primal-dual)
   * Gunakan barrier untuk ketaksamaan, selesaikan KKT perturbed; sangat efisien untuk masalah besar (LP/QP/Nonlin). Tidak memerlukan inisialisasi feasible biasanya.
4. Augmented Lagrangian / Method of Multipliers
   * Gabungan penalti dan Lagrange multipliers. Kuat secara numerik terhadap kesulitan penalti murni. Cocok bila SQP/interior-point sulit.
5. Projected gradient / proximal
   * Untuk constraint sederhana (kotak, simplex), lakukan langkah gradien lalu *project* kembali ke feasible set. Sederhana dan efisien bila projek cepat dihitung.
6. Solver praktis
   * Implementasi umum: SLSQP, trust-constr (SciPy), IPOPT, KNITRO, CVX/CVXOPT untuk kasus konveks. (Jika butuh contoh kode, saya bisa tambahkan.)

## 9. Kondisi orde kedua (sufficient conditions)

Setelah menemukan kandidat yang memenuhi KKT, untuk memastikan itu minimum lokal perlu cek orde kedua: Hessian Lagrangian

Cukupnya: pada semua arah yang memenuhi *linearized constraints* (yakni tangent space — arah yang tidak melanggar constraint aktif secara linear), harus berlaku

Jika ini terpenuhi → strict local minimum. Praktis: cek positif-definiteness pada reduced Hessian (Hessian dibatasi ke subruang tangensial).

## 10. Langkah sistematis menyelesaikan soal (untuk ujian / laporan)

1. Tuliskan masalah dan perjelas dan .
2. Periksa feasibility kandidat (apakah memenuhi kendala).
3. Bentuk Lagrangian .
4. Tulis kondisi KKT (stationarity, primal, dual, complementary slackness).
5. Selesaikan sistem aljabar (dapatkan ).
6. Periksa CQ (sebutkan bila LICQ terpenuhi).
7. Uji orde kedua (cek Hessian Lagrangian di ruang tangent) bila diminta.
8. Interpretasi: periksa apakah hasil masuk akal (boundary vs interior), berikan nilai fungsi & multipliers.

## 11. Contoh lengkap (langkah demi langkah — gabungan)

Soal: Minimalkan s.t. (dan bebas). Tuliskan kendala sebagai .

1. Lagrangian: .
2. KKT:
   * Primal: .
   * Dual: .
   * Stationarity:
   * Complementary slackness: .
3. Case analysis:
   * Jika → → primal feasibility? violates . Jadi impossible.
   * Jadi → complementary slackness forces (kendala aktif). Dengan dan → .
   * (positif). Sesuai.
4. Verifikasi orde kedua:
   * Hessian Lagrangian = $ ^2 f =
   * $ (tidak bergantung pada karena g linear).
   * Tangent space (direksi s.t. : dotted with = 0 → .
   * Untuk : . Jadi positif pada tangent → strict local minimum.

# Contoh 1: Fungsi Kuadrat dengan Kendala Linear

Masalah:

dengan kendala:

Interpretasi: Fungsi parabola (minimum di ), tetapi kendala memaksa solusi di . Solusi: .

# Contoh 2: Fungsi Dua Variabel dengan Equality Constraint

Masalah:

dengan kendala:

Interpretasi: Fungsi berbentuk mangkuk (minimum di ), tapi harus berada di garis . Solusi (dengan substitusi ):

Minimum di .

# Contoh 3: Fungsi Nonlinear dengan Inequality Constraint

Masalah:

dengan kendala:

Interpretasi: Fungsi tujuan mencari titik terdekat ke , tetapi harus berada dalam lingkaran . Solusi: Titik optimum berada pada tepi lingkaran (karena di luar lingkaran). Dengan metode proyeksi, solusi .

# Contoh 4: Fungsi dengan Beberapa Kendala Linear

Masalah:

dengan kendala:

Interpretasi: Fungsi linear, kendala linear → ini masalah *Linear Programming*. Solusi: Minimum terjadi di sudut feasible, yaitu di . Nilai fungsi: .

# Contoh 5: Fungsi Nonkonveks dengan Equality & Inequality

Masalah:

dengan kendala:

Interpretasi: Fungsi nonlinier nonkonveks; kendala equality membuat feasible set berupa garis segmen . Solusi: Substitusi :

Lalu cari minimum untuk . Dengan analisis turunan → diperoleh optimum sekitar .

# Pengimplementasian contoh-contoh fungsi tujuan tanpa kendala menggunakan coding program phyton

# Contoh 1: Fungsi Kuadrat dengan Kendala Linear

Masalah:

dengan kendala:

Coding program untuk fungsi tersebut:

import numpy as np

import matplotlib.pyplot as plt

# Fungsi

x = np.linspace(-5, 5, 400)

f = x\*\*2 + 4\*x + 4

plt.figure(figsize=(6,4))

plt.plot(x, f, label='f(x) = x² + 4x + 4')

plt.axvline(0, color='red', linestyle='--', label='Kendala: x ≥ 0')

plt.scatter(0, 4, color='blue', zorder=5, label='Solusi optimum')

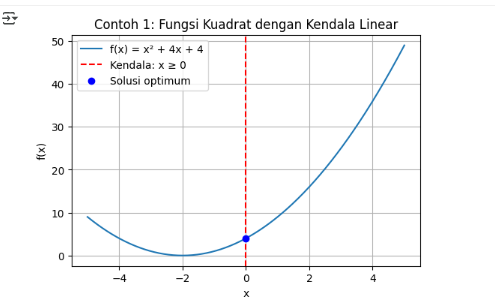
plt.title("Contoh 1: Fungsi Kuadrat dengan Kendala Linear")

plt.xlabel("x"); plt.ylabel("f(x)")

plt.legend(); plt.grid(True)

plt.show()

hasil coding program diatas:



# Contoh 2: Fungsi Dua Variabel dengan Equality Constraint

**Masalah:**

dengan kendala:

Coding program untuk fungsi tersebut:

import numpy as np

import matplotlib.pyplot as plt

# Grid

x = np.linspace(-1, 2, 200)

y = np.linspace(-1, 2, 200)

X, Y = np.meshgrid(x, y)

F = X\*\*2 + Y\*\*2

plt.figure(figsize=(6,6))

contours = plt.contour(X, Y, F, 20, cmap="viridis")

plt.clabel(contours, inline=True, fontsize=8)

plt.plot(x, 1-x, 'r--', label="Kendala: x+y=1")

plt.scatter(0.5,0.5, color='blue', label="Solusi optimum (0.5,0.5)")

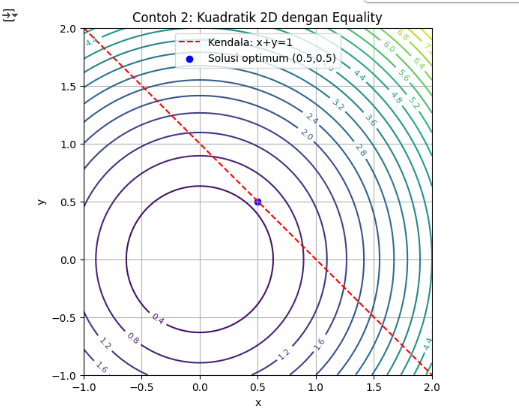
plt.title("Contoh 2: Kuadratik 2D dengan Equality")

plt.xlabel("x"); plt.ylabel("y")

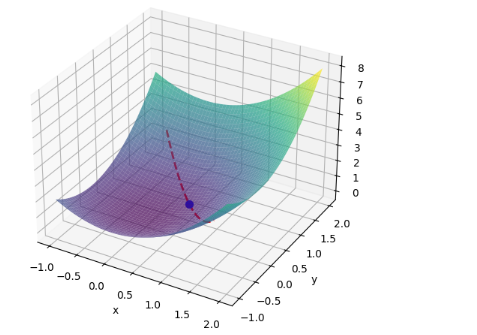
plt.legend(); plt.grid(True)

plt.show()

hasil coding program diatas:



Contoh dengan kuadratik 3D dengan kendala



# Contoh 3: Fungsi Nonlinear dengan Inequality Constraint

**Masalah:**

dengan kendala:

Coding program untuk fungsi tersebut:

import numpy as np

import matplotlib.pyplot as plt

# Grid

x = np.linspace(-1.5, 1.5, 300)

y = np.linspace(-1.5, 1.5, 300)

X, Y = np.meshgrid(x, y)

F = (X-1)\*\*2 + (Y-2)\*\*2

plt.figure(figsize=(6,6))

contours = plt.contour(X, Y, F, 20, cmap="viridis")

plt.clabel(contours, inline=True, fontsize=8)

# Kendala lingkaran

circle = plt.Circle((0,0), 1, color='red', fill=False, linestyle='--', label="x²+y² ≤ 1")

plt.gca().add\_patch(circle)

# Solusi (hasil proyeksi ke lingkaran)

plt.scatter(0.447,0.894, color='blue', label="Solusi optimum")

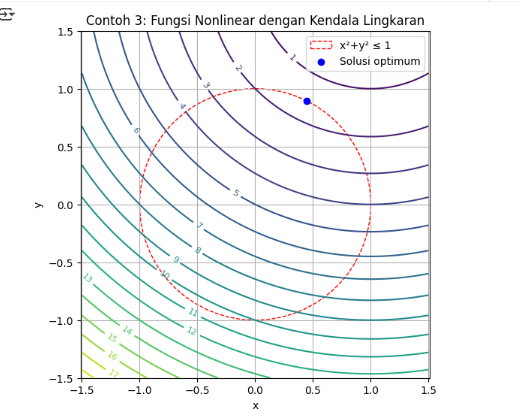
plt.title("Contoh 3: Fungsi Nonlinear dengan Kendala Lingkaran")

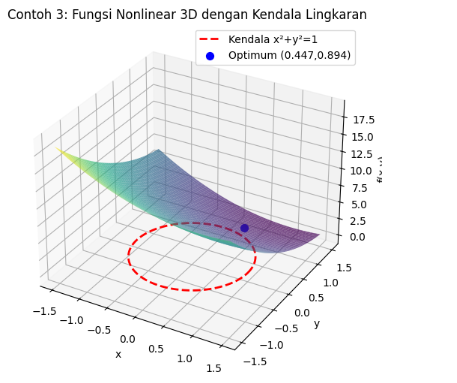
plt.xlabel("x"); plt.ylabel("y")

plt.legend(); plt.axis('equal'); plt.grid(True)

plt.show()

hasil program coding diatas:





# Contoh 4: Fungsi dengan Beberapa Kendala Linear

**Masalah:**

dengan kendala:

Coding program untuk fungsi tersebut:

import numpy as np

import matplotlib.pyplot as plt

# Grid untuk garis kendala

x = np.linspace(0, 5, 200)

y = 4 - x  # garis x+y=4

plt.figure(figsize=(6,6))

plt.plot(x, y, 'r--', label="Kendala: x+y=4")

plt.fill\_between(x, y, 6, where=(y>=0), color='lightgray', alpha=0.5, label="Daerah feasible")

plt.scatter(0,4, color='blue', label="Solusi optimum (0,4)")

plt.title("Contoh 4: Linear Programming")

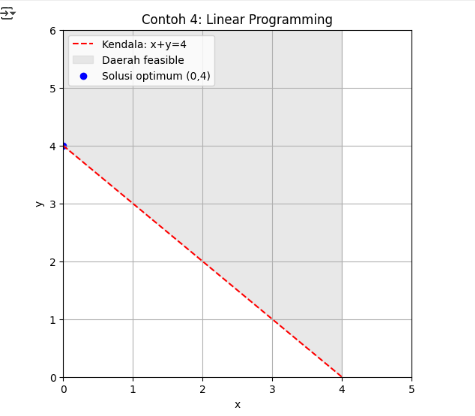
plt.xlabel("x"); plt.ylabel("y")

plt.xlim(0,5); plt.ylim(0,6)

plt.legend(); plt.grid(True)

plt.show()

Hasil coding program diatas:



# Contoh 5: Fungsi Nonkonveks dengan Equality & Inequality

**Masalah:**

dengan kendala:

Coding program untuk fungsi tersebut:

import numpy as np

import matplotlib.pyplot as plt

# Substitusi y=1-x

x = np.linspace(0,1,400)

y = 1-x

f = x\*\*4 + y\*\*4 - 2\*x\*y

plt.figure(figsize=(6,4))

plt.plot(x, f, label="f(x,1-x)")

plt.scatter(0.5,0.5\*\*4+0.5\*\*4-2\*0.5\*0.5, color='blue', label="Solusi optimum x=0.5,y=0.5")

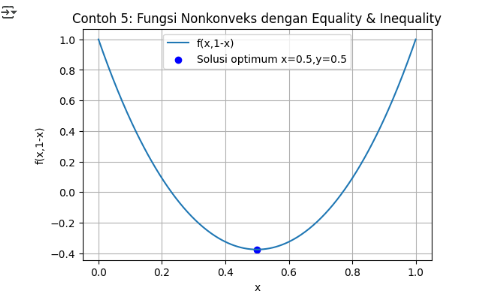
plt.title("Contoh 5: Fungsi Nonkonveks dengan Equality & Inequality")

plt.xlabel("x"); plt.ylabel("f(x,1-x)")

plt.legend(); plt.grid(True)

plt.show()

hasil coding program diatas:



# Analisis Optimasi dengan dan tanpa Kendala

1. **Optimasi tanpa Kendala**

## Gradien

Dalam matematika dan optimisasi, gradien (∇F) adalah vektor yang menunjukkan arah laju peningkatan terbesar dari suatu fungsi.

Untuk fungsi:

Gradiennya adalah:

∇F(x, y) = = (2x – 2, 2y - 4)

Pada setiap titik (x,y), vektor ini menunjuk ke arah 'mendaki' paling curam pada permukaan fungsi.

## Titik Kritis

Titik kritis adalah titik di mana gradien sama dengan vektor nol (∇F = 0⃗). Pada titik ini, tidak ada lagi arah yang dapat membuat nilai fungsi meningkat. Dengan demikian, kita berada pada puncak bukit atau dasar lembah.

Dengan menyetarakan gradien dengan nol:

2x – 2 = 0 ⟹ x = 1  
2y – 4 = 0 ⟹ y = 2

Maka, satu-satunya titik kritis fungsi ini adalah (1,2).

## Hessien dan Klasifikasi

Untuk menentukan sifat titik kritis (1,2), kita menggunakan matriks Hessien (H), yang terdiri dari turunan parsial kedua dari fungsi:

= 2  
 = 2  
 = 0

= 0

Sehingga, matriks Hessien adalah:

H = [[2, 0], [0, 2]]

Nilai determinannya: det(H)=(2)(2)−(0)(0)=4.

## Uji Turunan Kedua Multivariabel

• Jika det(H) > 0 dan ∂²F/∂x² > 0, maka titik kritis adalah minimum lokal.  
• Jika det(H) > 0 dan ∂²F/∂x² < 0, maka titik kritis adalah maksimum lokal.  
• Jika det(H) < 0, maka titik kritis adalah titik pelana.

Karena det(H) = 4 > 0 dan ∂²F/∂x² = 2 > 0, maka titik (1,2) adalah minimum lokal.

## Code dan Visualisasi

Hasil analitis ini dapat dikonfirmasi dengan visualisasi:  
1. Grafik Permukaan 3D menunjukkan bentuk paraboloid cekung ke atas, dengan titik terendah di dasarnya.  
2. Plot Kontur menampilkan elips konsentris yang semakin mengecil. Titik (1,2) terletak di pusat elips, yang menandakan posisi minimum.

# Fungsi tanpa kendala

import numpy as np

import matplotlib.pyplot as plt

from scipy.optimize import minimize

x = np.linspace(-1,3,200)

y = np.linspace(0,4,200)

X,Y = np.meshgrid(x,y)

F = X\*\*2 + Y\*\*2 - 2\*X - 4\*Y + 5

# Plot 3D

fig = plt.figure(figsize=(12,5))

ax = fig.add\_subplot(121, projection="3d")

ax.plot\_surface(X,Y,F,cmap="viridis")

ax.set\_title("Fungsi Tujuan Tanpa Kendala (Surface)")

ax.set\_xlabel("x"); ax.set\_ylabel("y"); ax.set\_zlabel("f")

# Contour + titik minimum

ax2 = fig.add\_subplot(122)

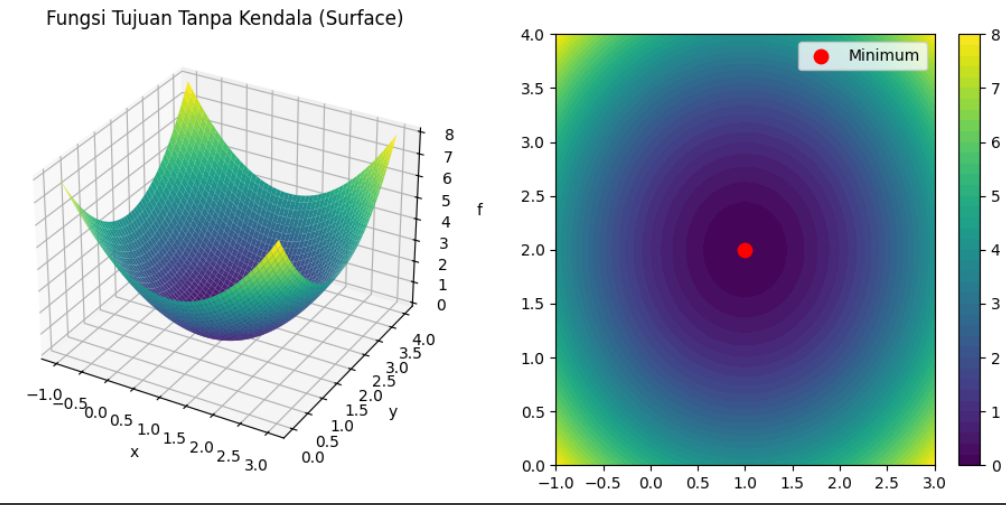
cp = ax2.contourf(X,Y,F,levels=40,cmap="viridis")

fig.colorbar(cp,ax=ax2)

ax2.scatter(1,2,c="red",s=80,label="Minimum")

ax2.legend()

plt.show()



1. **Optimasi dengan Kendala**

## Gradien: Arah Perubahan Fungsi

Fungsi yang ingin kita minimalkan adalah F(x,y) = x² + y². Gradiennya, yang menunjukkan arah peningkatan tercepat, adalah:

∇F(x, y) = = (2x, 2y)

Pada setiap titik (x, y), vektor gradien (2x, 2y) selalu menunjuk menjauhi titik asal (0, 0).

## Titik Kritis dengan Kendala

Tanpa kendala, titik kritis diperoleh saat ∇F = 0⃗, yaitu:

2x = 0 ⟹ x = 0  
2y = 0 ⟹ y = 0

Titik kritisnya adalah (0, 0), yang merupakan titik minimum global dari fungsi

F(x, y) = x² + y². Namun, kita harus mempertimbangkan kendala yang diberikan.

## Hessien dan Klasifikasi

Meskipun titik kritis (0, 0) tidak memenuhi kendala, kita tetap bisa menganalisisnya menggunakan matriks Hessien untuk menunjukkan bahwa itu memang titik minimum.

= 2  
 = 2  
 = 0

Matriks Hessien-nya adalah:

H = [[2, 0], [0, 2]]

Determinan Hessien-nya adalah det(H) = (2)(2) − (0)(0) = 4. Karena det(H) > 0 dan > 0, titik (0, 0) adalah minimum lokal.

## Metode Pengali Lagrange

Untuk menemukan titik minimum dari fungsi F(x, y) = x² + y² dengan kendala g(x, y) = x + y − 4 = 0, kita dapat menggunakan metode Pengali Lagrange. Metode ini memperkenalkan variabel baru λ (pengali Lagrange) dan membentuk fungsi baru yang disebut Fungsi Lagrange:

L(x, y, λ) = F(x, y) − λ g(x, y)  
L(x, y, λ) = x² + y² − λ(x + y − 4)

Untuk mencari titik kritis dari Fungsi Lagrange, kita ambil turunan parsial terhadap x, y, dan λ, lalu setarakan dengan nol:

∂L/∂x = 2x − λ = 0 ⟹ λ = 2x  
∂L/∂y = 2y − λ = 0 ⟹ λ = 2y  
∂L/∂λ = −(x + y − 4) = 0 ⟹ x + y = 4

Dari persamaan pertama dan kedua, kita dapatkan λ = 2x dan λ = 2y, sehingga 2x = 2y ⟹ x = y.

Kemudian, substitusikan x = y ke dalam persamaan kendala x + y = 4:  
2x = 4 ⟹ x = 2 ⟹ y = 2.

Jadi, titik kritis dari Fungsi Lagrange adalah (2, 2). Pada titik ini, nilai fungsi tujuan adalah F(2, 2) = 2² + 2² = 8.

## Ringkasan Analisis dengan Kendala

* Fungsi Tujuan: F(x, y) = x² + y² (meminimalkan jarak dari titik asal)
* Kendala: x + y = 4 dan x ≥ 0, y ≥ 0
* Metode: Pengali Lagrange
* Solusi Optimal: (2, 2)
* Nilai Minimum Fungsi: F(2, 2) = 8
* **Code Pyhton dan Visualisasi**

#Fungsi dengan kendala

def f(v):

    x,y = v

    return x\*\*2 + y\*\*2

cons = [{"type":"eq", "fun": lambda v: v[0] + v[1] - 4}]

bnds = [(0,None),(0,None)]

res = minimize(f, [1,3], constraints=cons, bounds=bnds)

# Grid untuk contour

x = np.linspace(0,4,200)

y = np.linspace(0,4,200)

X,Y = np.meshgrid(x,y)

F = X\*\*2 + Y\*\*2

fig = plt.figure(figsize=(12,5))

ax1 = fig.add\_subplot(121, projection="3d")

ax1.plot\_surface(X,Y,F,cmap="viridis",alpha=0.9)

ax1.set\_title("Fungsi dengan Kendala (Surface)")

ax2 = fig.add\_subplot(122)

cp = ax2.contourf(X,Y,F,levels=40,cmap="viridis")

fig.colorbar(cp,ax=ax2)

# garis kendala

ax2.plot(x,4-x,'w--',label="x+y=4")

# solusi optimum

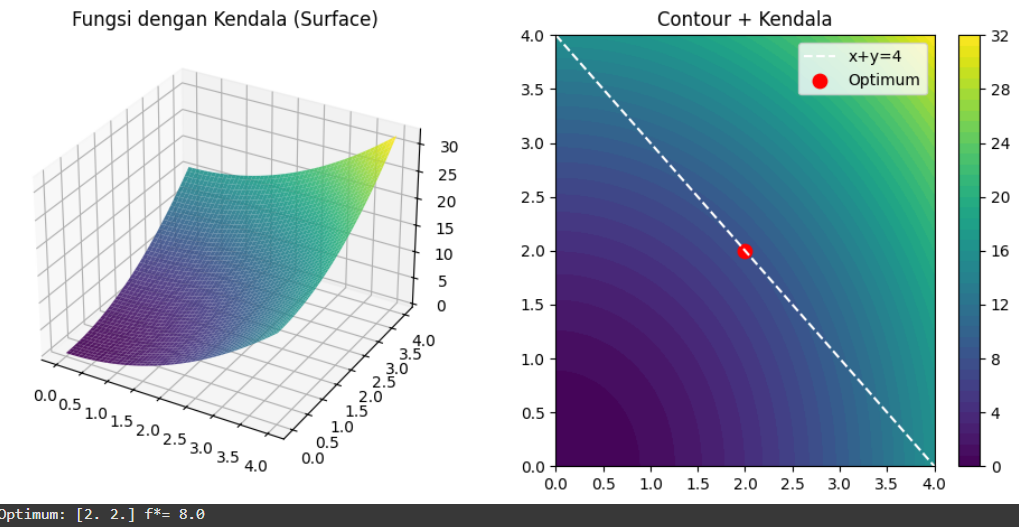
ax2.scatter(res.x[0],res.x[1],c="red",s=80,label="Optimum")

ax2.legend()

ax2.set\_title("Contour + Kendala")

plt.show()

print("Optimum:", res.x, "f\*=", res.fun)

****

**Fungsi Tujuan Tanpa Kendala**

1. Fungsi dan Gradien

Fungsi yang dianalisis adalah

Gradien fungsi ini adalah vector turunan parsial terhadap dan :

1. Mencari Titik Kritis

Untuk mencari titik kritis, setel gradien sama dengan nol :

Jadi, titik – titik kritisnya adalah

dan

1. Mencari Hessian Matriks dan klasifikasi

Matriks Hessian terdiri dari turunan parsial kedua :

* Untuk titik

Karena titik adalah **titik pelana.**

* Untuk titik

Karena dan , titik adalah **Minimum Lokal.**

**fungsi tujuan tanpa kendala dengan codingan program python:**

1. Fungsi

Codingan program fungsi tersebut :

import numpy as np

import sympy as sp

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

# (A) Fungsi tanpa kendala

x, y = sp.symbols('x y', real=True)

f = x\*\*3 - 3\*x + y\*\*2

# Gradien dan titik kritis

fx = sp.diff(f, x)

fy = sp.diff(f, y)

crit\_points = sp.solve([sp.Eq(fx, 0), sp.Eq(fy, 0)], [x, y], dict=True)

# Hessian

fxx = sp.diff(fx, x)

fyy = sp.diff(fy, y)

fxy = sp.diff(fx, y)

results\_A = []

for pt in crit\_points:

xv, yv = pt[x], pt[y]

H = sp.Matrix([[fxx.subs({x:xv,y:yv}), fxy.subs({x:xv,y:yv})],

[fxy.subs({x:xv,y:yv}), fyy.subs({x:xv,y:yv})]])

detH = H.det()

fxx\_val = fxx.subs({x:xv,y:yv})

if detH > 0 and fxx\_val > 0:

sifat = "Minimum lokal"

elif detH > 0 and fxx\_val < 0:

sifat = "Maksimum lokal"

elif detH < 0:

sifat = "Titik pelana"

else:

sifat = "Tidak dapat ditentukan"

results\_A.append((xv,yv,float(f.subs({x:xv,y:yv})),float(detH),float(fxx\_val),sifat))

# --------------------------------

# Plot 3D untuk (A)

fig = plt.figure(figsize=(8,5))

ax = fig.add\_subplot(111, projection='3d')

X = np.linspace(-2,2,200)

Y = np.linspace(-2,2,200)

Xg,Yg = np.meshgrid(X,Y)

Z = Xg\*\*3 - 3\*Xg + Yg\*\*2

ax.plot\_surface(Xg,Yg,Z,cmap='plasma',alpha=0.7)

ax.set\_title("f(x,y)=x^3-3x+y^2")

ax.set\_xlabel('x'); ax.set\_ylabel('y'); ax.set\_zlabel('f(x,y)')

for (xv,yv,fv,\_,\_,sifat) in results\_A:

ax.scatter(float(xv),float(yv),float(fv),color='red')

ax.text(float(xv),float(yv),float(fv),sifat)

plt.show()

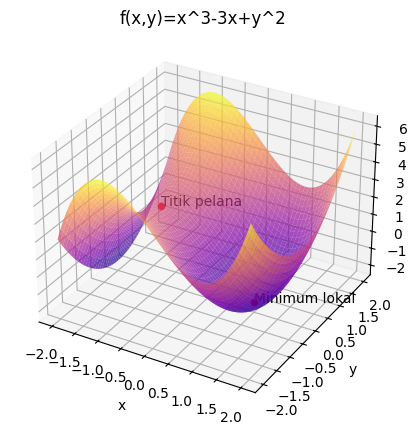
# --------------------------------

# Cetak hasil

print("=== A. Fungsi tanpa kendala ===")

for (xv,yv,fv,detH,fxx,sifat) in results\_A:

print(f"Titik: ({xv},{yv}), f={fv}, det(H)={detH}, fxx={fxx}, Sifat={sifat}")



Gambar 1. Fungsi Tujuan Tanpa Kendala

1. **Fungsi Tujuan dengan Kendala (Metode Lagrange)**
2. Fungsi dan Kendala yang di berikan adalah :

Dengan Kendala :

1. Fungsi Lagrange

Bentuk fungsi Lagrange

1. Persamaan Lagrange

Untuk mencari titik kritis :

1. Menyelesaikan Sistem Persamaan

Dari dan , Kita dapatkan

Substitusikan

Karena maka

Subtitusikan ke

Jadi titik kritisnya adalah

1. Interpretasi

* Titik adalah titik minimum dari pada garis kendala
* Nilai Minimumnya :

**fungsi tujuan dengan kendala codingan program python untuk 2 Dimensi:**

1. Fungsi dan Kendala yang di berikan adalah :

Dengan Kendala :

Codingan program fungsi tersebut :

import numpy as np

import sympy as sp

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

# --------------------------------

# (B) Fungsi dengan kendala: Lagrange

lam = sp.symbols('lam')

fB = x\*\*2 + y\*\*2

gB = x + y - 1

L = fB + lam\*gB

dLdx = sp.diff(L,x)

dLdy = sp.diff(L,y)

dLdlam = sp.diff(L,lam)

solB = sp.solve([dLdx,dLdy,dLdlam],[x,y,lam],dict=True)

# Kontur + kendala

fig2,ax2 = plt.subplots(figsize=(6,5))

X = np.linspace(-0.5,1.5,400)

Y = np.linspace(-0.5,1.5,400)

Xg,Yg = np.meshgrid(X,Y)

Z = Xg\*\*2+Yg\*\*2

cs = ax2.contour(Xg,Yg,Z,levels=20)

ax2.clabel(cs,inline=True,fontsize=8)

ax2.set\_title("f(x,y)=x^2+y^2 dengan kendala x+y=1")

ax2.set\_xlabel('x'); ax2.set\_ylabel('y')

xs = np.linspace(-0.5,1.5,100)

ys = 1 - xs

ax2.plot(xs,ys,'r--')

for s in solB:

xv,yv = s[x], s[y]

ax2.scatter(float(xv),float(yv),color='red')

ax2.text(float(xv),float(yv),"min")

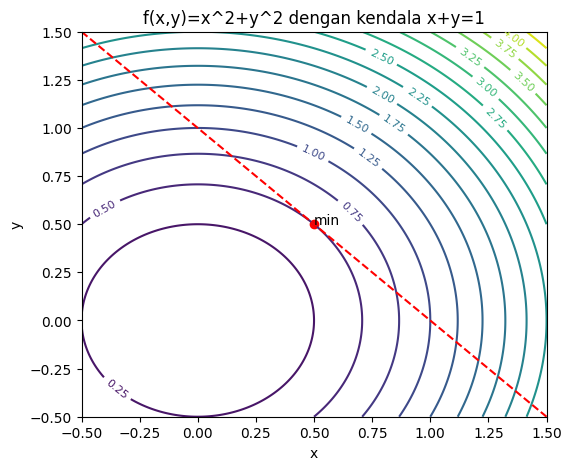
plt.show()

print("\n=== B. Fungsi dengan kendala ===")

for s in solB:

xv,yv = s[x],s[y]

print(f"Titik: ({xv},{yv}), f={fB.subs({x:xv,y:yv})}, lambda={s[lam]}")



Gambar 2. Fungsi Tujuan dengan kendala 2 Dimensi

**fungsi tujuan dengan kendala codingan program python untuk 3 Dimensi :**

import matplotlib.pyplot as plt

import numpy as np

from mpl\_toolkits.mplot3d import Axes3D

import sympy as sp

# Definisi simbol dan fungsi

x, y, lam = sp.symbols('x y lam')

fB = x\*\*2 + y\*\*2

gB = x + y - 1 # Constraint x + y - 1 = 0

# Buat data untuk plot 3D

X = np.linspace(-1.5, 1.5, 100)

Y = np.linspace(-1.5, 1.5, 100)

Xg, Yg = np.meshgrid(X, Y)

Z\_func = Xg\*\*2 + Yg\*\*2 # Surface of the function

# Data for the constraint plane x + y - 1 = 0

# We can represent this as y = 1 - x

# Let's create a meshgrid for x and z, and calculate y based on the constraint

X\_plane, Z\_plane = np.meshgrid(np.linspace(-1.5, 1.5, 50), np.linspace(0, 4, 50)) # Z range roughly based on function values

Y\_plane = 1 - X\_plane # Calculate Y based on the constraint

# Create 3D plot

fig = plt.figure(figsize=(10, 8))

ax = fig.add\_subplot(111, projection='3d')

# Plot permukaan fungsi

ax.plot\_surface(Xg, Yg, Z\_func, cmap='viridis', alpha=0.7, label='f(x,y) = x^2 + y^2')

# Plot bidang kendala (using X and Y as basis for grid)

# We need Z on this plane to represent the constraint in the context of the function's height

# A simpler way to visualize the constraint is to show the intersection or a plane at a constant Z, which is not correct here.

# A better way is to plot the plane itself.

ax.plot\_surface(X\_plane, Y\_plane, Z\_plane, color='red', alpha=0.3, rstride=5, cstride=5, label='Kendala x + y = 1')

# Titik minimum (dari solusi Lagrange)

# Gunakan solusiB dari sel sebelumnya

try:

from \_\_main\_\_ import solB

for s in solB:

xv, yv = s[x], s[y]

zv = float(fB.subs({x: xv, y: yv}))

ax.scatter(float(xv), float(yv), zv, color='red', s=100, zorder=5, label='Minimum (dengan kendala)')

except NameError:

print("Variabel 'solB' tidak ditemukan. Pastikan sel sebelumnya sudah dieksekusi.")

except KeyError as e:

print(f"KeyError: {e}. Make sure 'x' and 'y' are the correct symbolic keys in the dictionary.")

ax.set\_title("Permukaan f(x,y)=x^2+y^2 dan Bidang Kendala x+y=1")

ax.set\_xlabel('x')

ax.set\_ylabel('y')

ax.set\_zlabel('f(x,y)')

# Add a dummy artist for the constraint plane label if needed, since plot\_surface doesn't support label directly for legend

ax.legend()

plt.show()



Gambar 3. Fungsi tujuan dengan kendala 3 Dimensi

import numpy as np

import matplotlib.pyplot as plt

from numpy.linalg import eig, solve

# Fungsi dan gradien

def f(x):

    x1, x2 = x

    return 4\*x1\*\*2 + 2\*x1\*x2 + 5\*x2\*\*2 - 8\*x1 + 6\*x2

def grad(x):

    x1, x2 = x

    return np.array([8\*x1 + 2\*x2 - 8, 2\*x1 + 10\*x2 + 6])

# Hessian

H = np.array([[8, 2], [2, 10]])

# Titik stasioner

b = np.array([8, -6])

A = H.copy()

x\_star = solve(A, b)  # solve H x = b

print("Titik stasioner (x1,x2):", x\_star)

print("Nilai fungsi:", f(x\_star))

print("Eigenvalues Hessian:", eig(H)[0])

# Grid untuk visualisasi

x1 = np.linspace(-2, 3, 400)

x2 = np.linspace(-1, 3, 400)

X1, X2 = np.meshgrid(x1, x2)

Z = 4\*X1\*\*2 + 2\*X1\*X2 + 5\*X2\*\*2 - 8\*X1 + 6\*X2

# Plot Kontur 2D (Matplotlib)

plt.figure(figsize=(6,5))

CS = plt.contour(X1, X2, Z, levels=30, cmap='viridis')

plt.plot(x\_star[0], x\_star[1], 'ro', label='titik stasioner')

plt.title("Kontur f(x1,x2) — Contoh 1")

plt.xlabel("x1"); plt.ylabel("x2")

plt.legend()

plt.colorbar(CS)

plt.show()

# Plot 3D Surface (Matplotlib)

from mpl\_toolkits.mplot3d import Axes3D

fig = plt.figure(figsize=(8,6))

ax = fig.add\_subplot(111, projection='3d')

ax.plot\_surface(X1, X2, Z, cmap='coolwarm', alpha=0.8)

ax.scatter(x\_star[0], x\_star[1], f(x\_star), color='red', s=50, label='titik stasioner')

ax.set\_xlabel('x1'); ax.set\_ylabel('x2'); ax.set\_zlabel('f(x1,x2)')

plt.title('3D Surface f(x1,x2) — Contoh 1')

plt.legend()

plt.show()

# VERSI INTERAKTIF (PLOTLY)

import plotly.graph\_objects as go

# Plot interactive 3D dengan Plotly

fig = go.Figure(data=[go.Surface(z=Z, x=X1, y=X2, colorscale="Viridis", opacity=0.9)])

fig.add\_trace(go.Scatter3d(

    x=[x\_star[0]], y=[x\_star[1]], z=[f(x\_star)],

    mode='markers',

    marker=dict(size=6, color='red'),

    name="Titik Stasioner"

))

fig.update\_layout(

    title="3D Surface f(x1, x2) (Interactive)",

    scene=dict(

        xaxis\_title="x1",

        yaxis\_title="x2",

        zaxis\_title="f(x1,x2)"

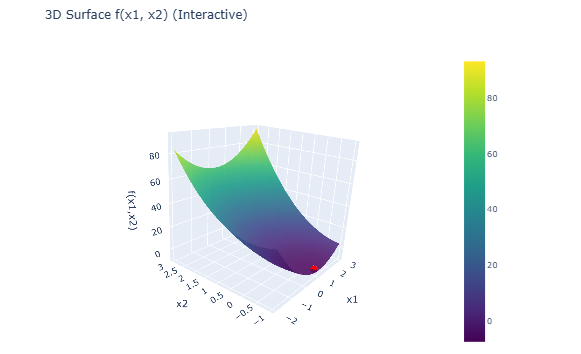
    ),

    width=700,

    height=600

)

fig.show()



Penjelesan:

Fungsi tujuan tanpa kendala

Fungsi tujuan:

1. **Gradien**

Gradien menunjukkan arah kenaikan tercepat fungsi. Titik stasioner diperoleh saat .

1. **Titik stasioner — selesaikan sistem**

Selesaikan sistem linear:

1. **Solusi aljabar (aturan Cramer)**

Determinant matriks koefisien:

Pembilang untuk (ganti kolom pertama dengan vektor kanan):

Sehingga

Pembilang untuk (ganti kolom kedua dengan vektor kanan):

Sehingga

Jadi titik stasioner:

1. **Hessian dan eigenvalues**

Minor utama pertama: . Determinan Hessian:

Eigenvalues (untuk informasi):

Karena semua minor utama positif positif definite, sehingga titik stasioner adalah **minimum global**.

1. **Nilai fungsi di titik stasioner**

Substitusi

Hitung setiap suku (penyebut sama ):

Jumlahkan semua :

# Import library

import numpy as np

import matplotlib.pyplot as plt

from sympy import symbols, diff, solve, Matrix

# Simbol

x1, x2 = symbols('x1 x2')

# Fungsi

f2 = 2\*x1\*\*2 + 5\*x1\*x2 + 3\*x2\*\*2 -6\*x1 +4\*x2

# Gradien

grad\_f2 = [diff(f2, x1), diff(f2, x2)]

print("Gradien Contoh 2:", grad\_f2)

# Titik stasioner

sol2 = solve(grad\_f2, (x1, x2))

print("Titik stasioner Contoh 2:", sol2)

# Hessian

H2 = Matrix([[diff(f2,x1,x1), diff(f2,x1,x2)],

             [diff(f2,x2,x1), diff(f2,x2,x2)]])

print("Hessian Contoh 2:")

print(H2)

# Nilai fungsi di titik stasioner (fix)

val2 = f2.subs(sol2)

print("Nilai fungsi di titik stasioner Contoh 2:", val2)

# Visualisasi 3D

x = np.linspace(-60,60,100)

y = np.linspace(-10,60,100)

X,Y = np.meshgrid(x,y)

Z2 = 2\*X\*\*2 + 5\*X\*Y + 3\*Y\*\*2 -6\*X +4\*Y

fig = plt.figure(figsize=(8,6))

ax = fig.add\_subplot(111, projection='3d')

ax.plot\_surface(X,Y,Z2, cmap='viridis', alpha=0.8)

ax.set\_xlabel('x1'); ax.set\_ylabel('x2'); ax.set\_zlabel('f2(x1,x2)')

plt.title('3D Plot Contoh 2')

plt.show()

# Contour

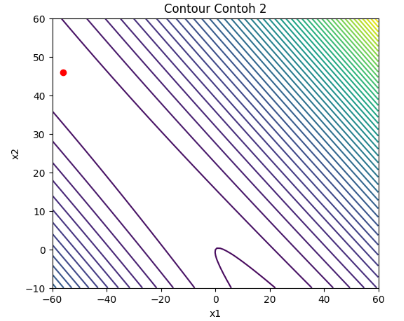
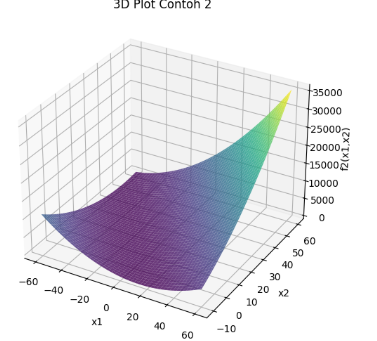
plt.figure(figsize=(6,5))

plt.contour(X,Y,Z2,50, cmap='viridis')

plt.scatter(float(sol2[x1]), float(sol2[x2]), color='red')

plt.xlabel('x1'); plt.ylabel('x2'); plt.title('Contour Contoh 2')

plt.show()



Penjelasan:

Fungsi tujuan tanpa kendala

Fungsi tujuan:

1. **Gradien**
2. **Titik Stasioner**

Selesaikan sistem:

1. **Solusi Aljabar (aturan Cramer)**

Pembilang untuk () (ganti kolom pertama dengan vektor kanan):

Sehingga .

Pembilang untuk () (ganti kolom kedua dengan vektor kanan):

Sehingga .

Jadi titik stasioner:

1. **Hessian**

5. **Klasifikasi (maks/min/saddle)**

Minor utama pertama:

Determinan Hessian:  
Karena , Hessian indefinite titik stasioner adalah **saddle point**.

1. **Nilai fungsi di titik stasioner**

Substitusi ke fungsi :

Jadi nilai fungsi di titik stasioner : .

# Import library

import numpy as np

import matplotlib.pyplot as plt

from sympy import symbols, diff, solve, Matrix

# Simbol

x1, x2 = symbols('x1 x2')

# Fungsi

f3 = 6\*x1\*\*2 + 2\*x1\*x2 + 4\*x2\*\*2 -12\*x1 +3\*x2

# Gradien

grad\_f3 = [diff(f3, x1), diff(f3, x2)]

print("Gradien Contoh 3:", grad\_f3)

# Titik stasioner

sol3 = solve(grad\_f3, (x1, x2))

print("Titik stasioner Contoh 3:", sol3)

# Hessian

H3 = Matrix([[diff(f3,x1,x1), diff(f3,x1,x2)],

             [diff(f3,x2,x1), diff(f3,x2,x2)]])

print("Hessian Contoh 3:")

print(H3)

# Nilai fungsi di titik stasioner (fix)

val3 = f3.subs(sol3)

print("Nilai fungsi di titik stasioner Contoh 3:", val3)

# Visualisasi 3D

x = np.linspace(-2,6,100)

y = np.linspace(-2,2,100)

X,Y = np.meshgrid(x,y)

Z3 = 6\*X\*\*2 + 2\*X\*Y + 4\*Y\*\*2 -12\*X +3\*Y

fig = plt.figure(figsize=(8,6))

ax = fig.add\_subplot(111, projection='3d')

ax.plot\_surface(X,Y,Z3, cmap='coolwarm', alpha=0.8)

ax.set\_xlabel('x1'); ax.set\_ylabel('x2'); ax.set\_zlabel('f3(x1,x2)')

plt.title('3D Plot Contoh 3')

plt.show()

# Contour

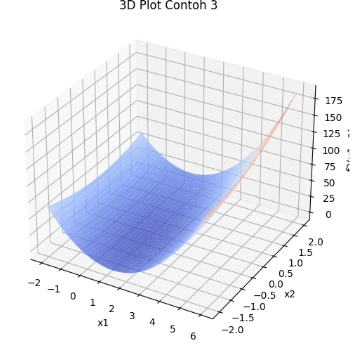
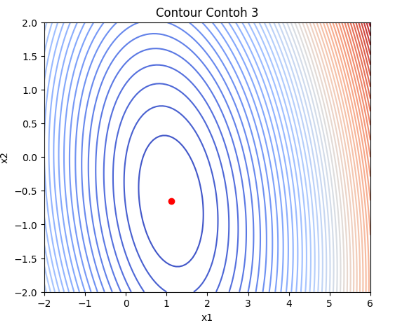
plt.figure(figsize=(6,5))

plt.contour(X,Y,Z3,50, cmap='coolwarm')

plt.scatter(float(sol3[x1]), float(sol3[x2]), color='red')

plt.xlabel('x1'); plt.ylabel('x2'); plt.title('Contour Contoh 3')

plt.show()



Penjelasan:

Fungsi tujuan tanpa kendala

Fungsi tujuan:

1. **Gradien**
2. **Titik Stasioner**

Selesaikan sistem:

1. **Solusi Aljabar**

Determinant:

Hitung :

Hitung :

Titik stasioner: .

1. **Hessian**

5. **Klasifikasi (maks/min)**

Minor pertama:

Determinan Hessian:  
 Hessian positif definite minimum global.

6. **Nilai fungsi di titik stasioner**

Titik stasioner yang diperoleh adalah:

Substitusi ke fungsi tujuan:

Hitung setiap suku:

Jumlahkan semuanya:

Hasil akhir:

Sederhanakan:

# Import library

import numpy as np

import matplotlib.pyplot as plt

from sympy import symbols, diff, solve, Matrix

# Simbol

x1, x2, lam = symbols('x1 x2 lam')  # tambahkan lambda untuk Lagrangian

# Fungsi tujuan

f = x1\*\*2 + x2\*\*2 - 2\*x1 + 2\*x2

# Kendala

g = x1 + x2 - 1

# Lagrangian

L = f + lam\*g

# Gradien Lagrangian

grad\_L = [diff(L, x1), diff(L, x2), diff(L, lam)]

print("Gradien Lagrangian:", grad\_L)

# Titik stasioner (x1, x2, λ)

sol = solve(grad\_L, (x1, x2, lam))

print("Solusi (x1, x2, λ):", sol)

# Ambil solusi x1\*, x2\*

x1\_star, x2\_star, lam\_star = sol[x1], sol[x2], sol[lam]

print("Titik stasioner:", (x1\_star, x2\_star))

# Nilai fungsi di titik stasioner

val = f.subs({x1:x1\_star, x2:x2\_star})

print("Nilai fungsi di titik stasioner:", val)

# Hessian fungsi tujuan

H = Matrix([[diff(f,x1,x1), diff(f,x1,x2)],

            [diff(f,x2,x1), diff(f,x2,x2)]])

print("Hessian f:")

print(H)

# --- Visualisasi ---

# Grid untuk plot

x = np.linspace(-1, 3, 200)

y = np.linspace(-2, 2, 200)

X, Y = np.meshgrid(x, y)

Z = X\*\*2 + Y\*\*2 - 2\*X + 2\*Y

# 3D Plot

from mpl\_toolkits.mplot3d import Axes3D

fig = plt.figure(figsize=(8,6))

ax = fig.add\_subplot(111, projection='3d')

ax.plot\_surface(X, Y, Z, cmap='coolwarm', alpha=0.8)

# Plot garis kendala x1+x2=1

x\_line = np.linspace(-1, 3, 100)

y\_line = 1 - x\_line

ax.plot(x\_line, y\_line, f.subs({x1:x\_line[0], x2:y\_line[0]}), 'r--', label="Kendala")

# Titik stasioner

ax.scatter(float(x1\_star), float(x2\_star), float(val), color='red', s=50)

ax.set\_xlabel('x1'); ax.set\_ylabel('x2'); ax.set\_zlabel('f(x1,x2)')

plt.title("3D Plot Fungsi dengan Kendala")

plt.show()

# Contour Plot

plt.figure(figsize=(6,5))

plt.contour(X,Y,Z,30, cmap='coolwarm')

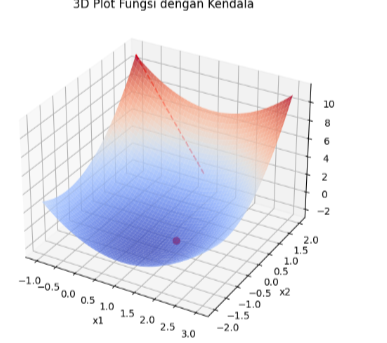
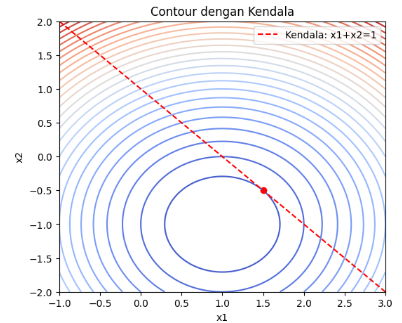
plt.plot(x\_line, y\_line, 'r--', label="Kendala: x1+x2=1")

plt.scatter(float(x1\_star), float(x2\_star), color='red')

plt.xlabel('x1'); plt.ylabel('x2'); plt.title('Contour dengan Kendala')

plt.legend()

plt.show()



Fungsi tujuan dengan kendala

Diberikan fungsi tujuan:

dengan kendala linear:

1. **Lagrangian dan Gradien Lagrangian**

Bentuk Lagrangian:

Gradien Lagrangian:

Titik stasioner pada kendala diperoleh dengan menyelesaikan .

1. **Sistem Persamaan (gradien Lagrangian = 0)**

Persamaan:

Dari persamaan pertama: .  
Dari persamaan kedua: .  
Samakan kedua ekspresi untuk :

Dengan kendala , selesaikan sistem:

Menjumlahkan kedua persamaan:  
.  
Substitusi ke kendala:  
.  
Diperoleh .

Jadi solusi lengkap:

Titik stasioner (pada kendala):

1. **Nilai fungsi di titik stasioner**

Substitusi ke menghasilkan:

1. **Hessian fungsi tujuan dan klasifikasi**

Hessian fungsi tujuan :

Minor utama pertama = 2 > 0.

Determinan = .

Eigenvalue = 2 > 0.  
Ruang tangen (gradien kendala ):

Kuadratik bentuk pada ruang tangen:

Positif definite pada ruang tangen minimum terikat.

1. **Jawaban akhir**

Titik stasioner: , , **Minimum terikat**.

Dalam analisis optimasi, fungsi tujuan tanpa kendala sering digunakan untuk menggambarkan suatu sistem atau proses yang ingin dicapai nilai ekstremnya, baik berupa minimum maupun maksimum. Pada kasus ini, saya akan meninjau fungsi kuadrat sederhana dua variabel:

Fungsi ini dipilih karena bersifat dasar, namun mewakili bentuk umum fungsi kuadrat yang banyak dijumpai dalam bidang matematika terapan, fisika, maupun ekonomi.

* **Teori Dasar**
* **Gradien**

Gradien merupakan vektor yang berisi turunan parsial pertama dari fungsi terhadap tiap variabel. Titik kritis terjadi jika gradien sama dengan nol.

* **Matriks Hessian** (**H**)

Hessian adalah matriks yang berisi turunan parsial kedua, digunakan untuk mengklasifikasikan titik kritis.

* **Kriteria Klarifikasi**

Dengan menggunakan determinan Hessian (DDD), titik kritis diklasifikasikan sebagai berikut:

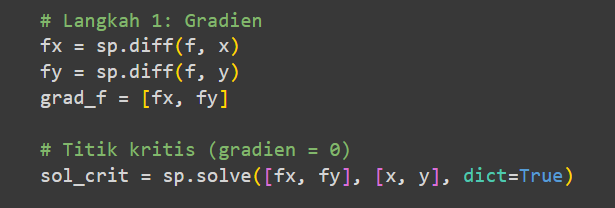
* Jika dan minimum lokal.
* Jika dan maksimum lokal.
* Jika titik pelana.
* Jika tidak dapat ditentukan.
* **Langkah-langkah Penyelesaian**
  + **Menentukan Gradien**

Fungsi:

Turunan Parsial:

Gradien:

**Potongan kode:**



* + **Menentukan Titik Kritis**

Syarat gradien = 0

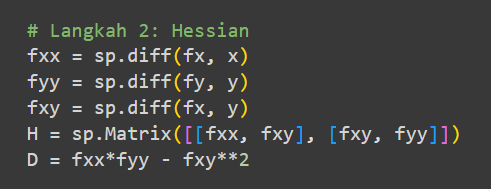
Jadi, titik kritis terdapat pada koordinat pusat (0,0).

* + **Matriks Hessian**

Turunan kedua:

Maka:

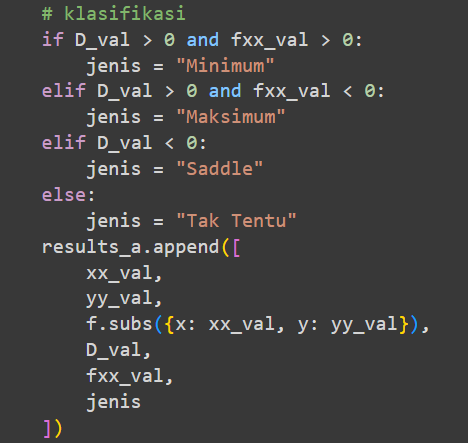
**Potongan kode:**



* + **Evaluasi Determinan Hessian**

Karena dan maka titik adalah minimum lokal.

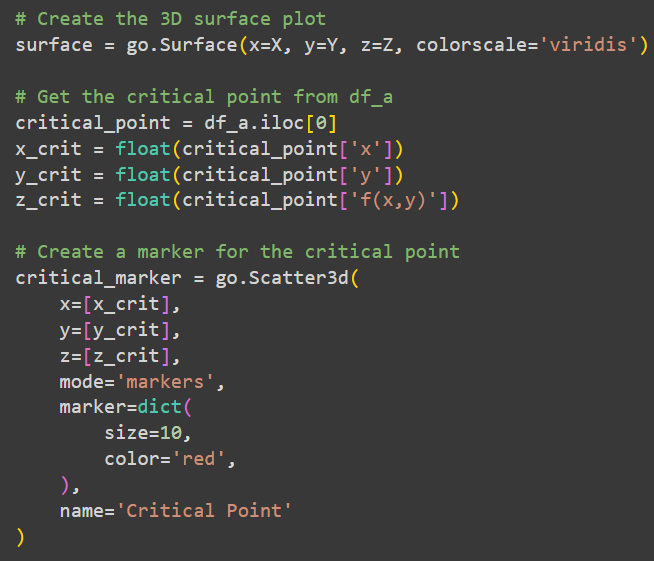
**Potongan kode:**

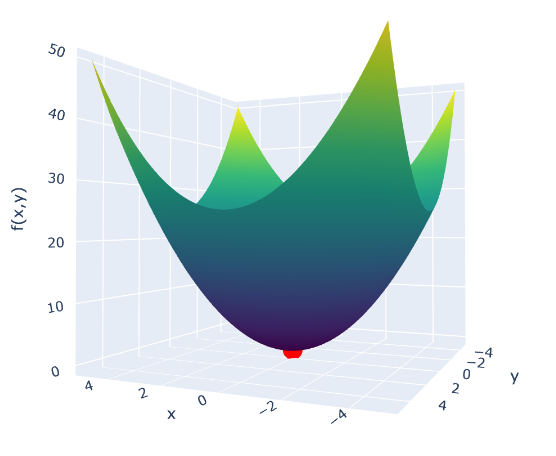
****

* **Hasil dan Interpretasi**
* Titik kritis:
* Nilai fungsi pada titik kritis:
* Klasifikasi titik kritis: **minimum lokal**

Secara geometris, grafik membentuk permukaan **paraboloid cekung di atas**, dengan puncak minimum di pusat koordinat.

**Potongan kode:**





* **Kesimpulan**

Dari analisis yang dilakukan, fungsi hanya memiliki satu titik kritis di . Berdasarkan evaluasi Hessian, titik tersebut merupakan **minimum lokal** dengan nilai fungsi minimum Hal ini sesuai dengan bentuk fungsi kuadrat yang selalu positif dan berbentuk paraboloid.

**Analisis Fungsi Tujuan Dengan Kendala**

* **Pendahuluan**

Dalam optimasi, seringkali sebuah fungsi tujuan tidak dapat dievaluasi bebas pada seluruh domain, melainkan harus memenuhi syarat tertentu (kendala). Salah satu metode yang umum digunakan untuk menyelesaikan masalah seperti ini adalah **Metode Lagrange Multiplier**.

Pada kasus ini, saya akan menganalisis fungsi tujuan:

Dengan kendala berupa lingkaran satuan:

Tujuannya adalah menentukan titik-titik kritis dari di bawah kendala tersebut, serta mengklasifikasikannya sebagai titik maksimum atau minimum.

* **Teori Dasar**
* **Metode Lagrange Multiplier**

Untuk fungsi tujuan dengan kendala , dibentuk fungsi lagrange:

Dengan disebut pengali Lagrange.

* **Syarat Titik Kritis**

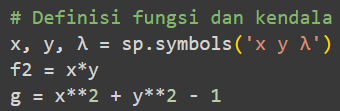
Titik kritis diperoleh dengan menyelesaikan sistem persamaan:

* **Interpretasi**

Titik optimum dicapai ketika gradien fungsi tujuan sejajar dengan gradien kendala. Secara geometris, ini berarti garis level fungsi tujuan menyinggung kurva kendala.

* **Langkah-Langkah Penyelesaian**
  + **Formulasi Fungsi Lagrange:**

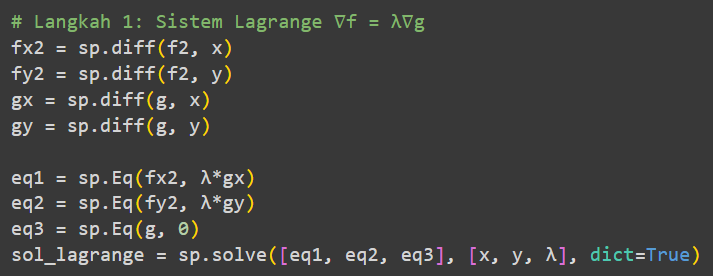
**Potongan kode:**



* + **Hitung Turunan Parsial**

Turunan terhadap masing-masing variabel:

**Potongan kode:**



* + **Sistem Persamaan**

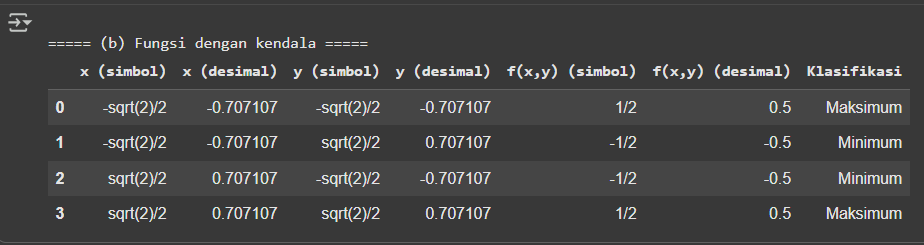
Syarat titik kritis diperoleh dari:

**Potongan kode:**

****

* + **Solusi Titik Kritis**

Hasil penyelesaian sistem memberikan empat titik kritis:



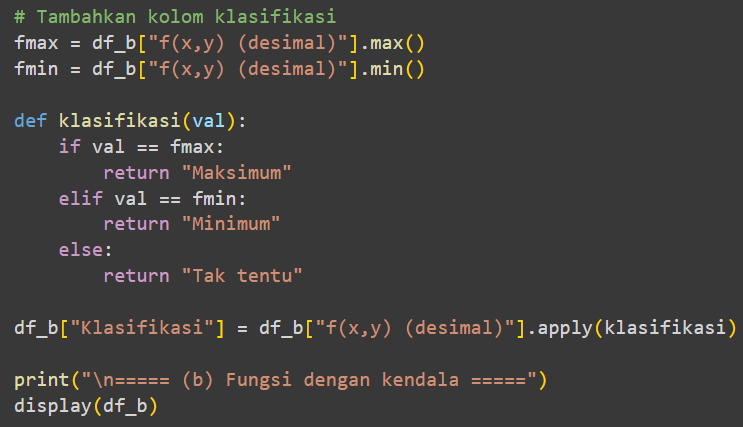
* **Evaluasi Fungsi Tujuan di Titik Kritis**

Nilai fungsi di setiap titik kritis:

Dari sini terlihat bahwa:

* Dua titik menghasilkan nilai maksimum
* Dua titik lainnya menghasilkan nilai minimum

**Potongan kode:**

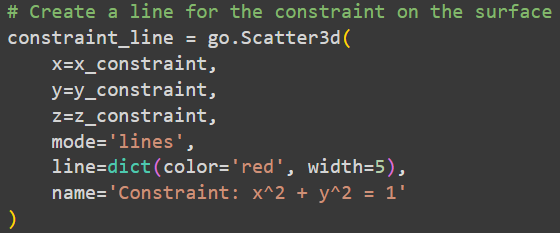
****

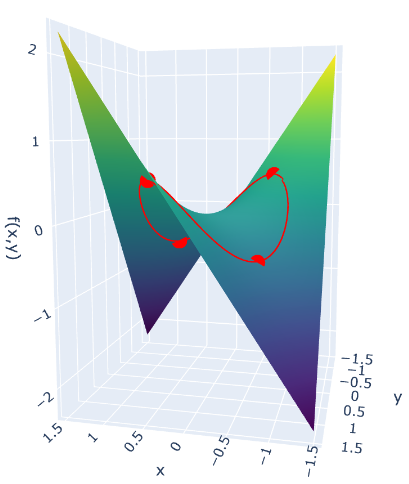
* **Visualisasi**

Secara geometris:

* Fungsi membentuk permukaan pelana.
* Kendala adalah lingkaran satuan di bidang .
* Titik kritis adalah titik pada lingkaran satuan dimana garis level menyinggung lingkaran tersebut.

**Potongan kode:**





* **Kesimpulan**

Dari analisis menggunakan metode Lagrange Multiplier, diperoleh:

* Empat titik kritis di lingkaran satuan.
* Dua titik merupakan **titik maksimum** dengan nilai
* Dua titik merupakan **titik minimum** dengan nilai

Kendala lingkaran membatasi pergerakan menuju asal, sehingga nilai maksimum/minimum tercapai pada titik di mana gradien fungsi tujuan sejajar dengan gradien kendala.

* **Fungsi Tujuan Tanpa Kendala**

Teori → Fungsi tujuan tanpa kendala berarti kita ingin mencari titik optimum (minimum atau maksimum) dari suatu fungsi **tanpa syarat tambahan**. Artinya, pencarian titik optimum dilakukan hanya berdasarkan sifat internal fungsi itu sendiri.

Langkah-langkah penyelesaiannya adalah sebagai berikut:

* **Hitung Gradien**

Gradien dari fungsi dituliskan sebagai vektor turunan parsial:

Titik stasioner (kandidat optimum) diperoleh dengan menyelesaikan persamaan

Secara geometris, gradien pada suatu titik menunjukkan arah kenaikan tercepat fungsi; titik stasioner adalah tempat di mana vektor gradien bernilai nol (tidak ada arah kenaikan/penurunan lokal).

* **Cari Hessian**

Setelah menemukan titik stasioner, kita menghitung matriks Hessian untuk menentukan sifat titik tersebut (minimum lokal, maksimum lokal, saddle point). Hessian didefinisikan sebagai:

* **Analisis Hessian**
* Jika positif definit (semua nilai eigen ) maka titik tersebut adalah minimum lokal (jika fungsi kuadrat/kuadratik dengan Hessian konstan, ini juga minimum global).
* Jika negatif definit → maksimum lokal.
* Jika indefinit (positif dan negatif campur → saddle point.

Karena Hessian untuk fungsi kuadrat seringkali konstan, analisis ini langsung menentukan sifat global ketika fungsi adalah fungsi kuadrat positif definit (konveks).

**Contoh:**

Diberikan fungsi

Langkah 1-Gradien:

Hitung turunan parsial:

Syarat memberikan sistem:

Maka diperoleh:

Jadi titik stasioner kandidat adalah

Langkah 2-Hessian:

Hessian fungsi ini adalah matriks konstan

Nilai eigen Hessian adalah keduanya positif. Jadi positif definit.

Kesimpulan sifat titik:

Karena Hessian positif definit, titik adalah minimum lokal. Karena fungsi ini kuadrat dengan Hessian positif definit di seluruh titik ini juga merupakan minimum global.

Langkah 3 — Nilai fungsi di titik optimum (hitung digit per digit):

Hitung langkah demi langkah:

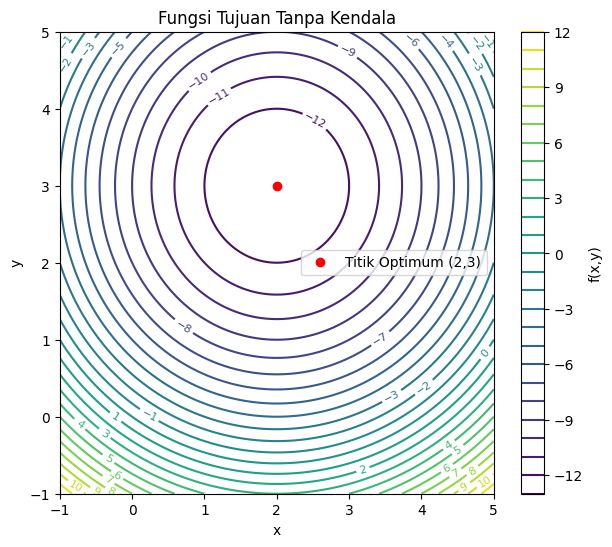
Sehingga

Jadi nilai minimum globalnya adalah

Completing the square (penjelasan alternatif yang memperjelas pusat paraboloid):

Kita dapat menulis ulang fungsi sebagai:

Dari bentuk ini jelas terlihat pusat paraboloid pada dan nilai minimum .



* Plot kontur 2D: Garis-garis kontur adalah lingkaran konsentris yang berpusat pada Masing-masing kontur merepresentasikan level set Nilai semakin besar ketika kita bergerak menjauh dari pusat Titik merah pada gambar menandai posisi optimum
* Plot 3D: Permukaan adalah sebuah “mangkuk” (paraboloid) terbuka ke atas dengan puncak terendah di Karena kurvatur sama di arah dan (Hessian diagonal dengan entri sama), bentuknya simetris di sekitar pusat.

Makna gradien pada plot: Gradien di titik mana pun menunjukkan arah tercepat kenaikan nilai fungsi; pada titik pusat gradien yang konsisten dengan titik minimum. Nilai Hessian (melalui elemen diagonal ) memberi tahu seberapa “curam” mangkuk tersebut semakin besar angka pada Hessian, semakin curam kurvaturnya.

* **Fungsi Tujuan Dengan Kendala**

Teori → Fungsi tujuan dengan kendala berarti kita mencari titik optimum dari fungsi dengan adanya syarat tambahan (constraint). Kendala bisa berupa persamaan maupun pertidaksamaan

Metode yang umum digunakan adalah **Metode Lagrange** (untuk kendala persamaan).

Langkah-langkah penyelesaian:

* **Bentuk Fungsi Lagrange**

Untuk masalah:

dengan kendala

bentuk Lagrangian:

Dimana adalah multiplier Lagrange.

* **Turunkan Fungsi Lagrange**

Hitung turunan parsial dan set sama nol:

Persamaan terakhir menghasilkan kembali kendala

* **Selesaikan Sistem Persamaan**

Dari ketiga persamaan di atas diperoleh kandidat titik optimum serta nilai Karena kita sedang bekerja dengan fungsi kuadrat dan kendala linear (affine), solusi adalah unik dan merupakan solusi global apabila konveks.

* **Analisis Hasil**

Periksa apakah titik yang diperoleh memenuhi kondisi kendala dan analisis apakah itu minimum atau maksimum. Untuk fungsi konveks dan kendala affine, kondisi Lagrange bersifat cukup untuk memastikan solusi global.

**Contoh:**

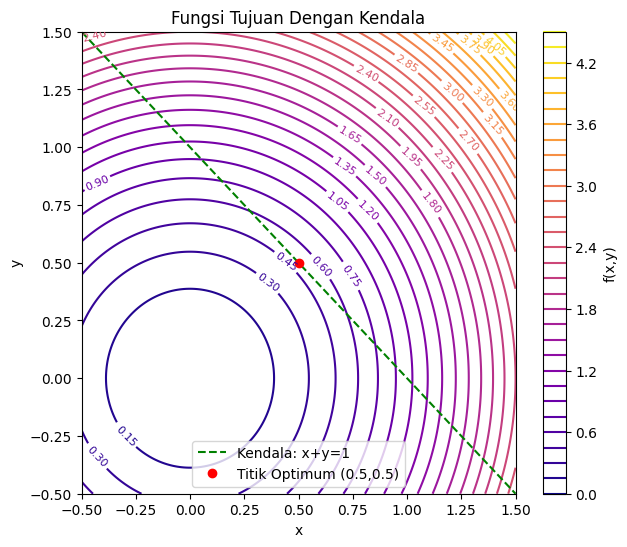
Pertimbangkan:

dengan kendala

Metode Lagrange — Langkah demi langkah

* Bentuk Lagrangian:
* Turunan parsial:
* Dari persamaan pertama dan kedua diperoleh:
* Gunakan kendala karena kita punya sehingga:
* Nilai multiplier dari
* Nilai fungsi di titik tersebut:

Interpretasi: Titik adalah titik optimum (minimum) pada garis kendala



* Pada gambar kontur untuk fungsi garis-garis kontur adalah lingkaran berpusat di. Garis kendala digambarkan sebagai garis lurus (biasanya digambar dengan garis putus-putus hijau). Titik optimum berada pada garis hijau tersebut di tempat lingkaran terkecil yang masih menyentuh garis: itu adalah titik kontak (titik proyek) antara lingkaran yang berpusat di origin dan garis kendala.
* Pada plot 3D, permukaan berbentuk mangkuk simetris (bowl). Garis kendala memotong mangkuk tersebut; titik minimum di garis kendala adalah posisi pada garis itu yang berada pada ketinggian terendah pada permukaan.

**Pentingnya Optimasi Nonlinear dalam Matematika**

Dalam Prodi Matematika, optimasi nonlinear dipandang sangat penting karena:

1. **Landasan teoritis** → memberikan pemahaman mendalam mengenai gradien, Hessian, serta titik kritis, yang merupakan kelanjutan dari Kalkulus Multivariabel.
2. **Keterampilan analitis** → melatih mahasiswa berpikir sistematis dalam menyelesaikan masalah matematis yang kompleks.
3. **Keterampilan komputasi** → membiasakan mahasiswa menggunakan perangkat lunak seperti Python, MATLAB, atau Maple untuk menyelesaikan masalah numerik yang sulit diselesaikan manual.
4. **Aplikasi luas** → menjembatani matematika murni dengan berbagai bidang terapan, mulai dari ekonomi, teknik, sains, hingga kecerdasan buatan.

**Penerapan Optimasi Nonlinear dalam Kehidupan Nyata**

Beberapa contoh penggunaan optimasi nonlinear:

* **Ekonomi dan Manajemen:**  
  Perusahaan sering dihadapkan pada masalah penentuan kombinasi produksi yang menghasilkan keuntungan maksimal dengan biaya minimal. Fungsi keuntungan atau biaya sering bersifat nonlinear karena adanya faktor skala produksi dan perubahan harga.
* **Teknik dan Rekayasa:**  
  Dalam perancangan jembatan atau gedung, insinyur harus mencari bentuk struktur yang **kuat tetapi hemat bahan**. Fungsi yang menggambarkan tegangan dan energi struktur biasanya berbentuk nonlinear.
* **Fisika dan Ilmu Alam:**  
  Banyak fenomena alam, seperti getaran dan gelombang, dimodelkan dengan fungsi sinus dan cosinus. Menentukan **energi minimum** atau **posisi setimbang** suatu sistem adalah contoh langsung dari optimasi nonlinear.
* **Ilmu Komputer dan Machine Learning:**  
  Hampir semua algoritma machine learning melibatkan proses optimasi nonlinear. Misalnya, pelatihan jaringan saraf tiruan bertujuan meminimalkan fungsi loss yang rumit dan bersifat nonlinear.

**Tujuan Pembelajaran Optimasi Nonlinear di Prodi Matematika**

Dalam konteks pendidikan, khususnya Prodi Matematika, tujuan mempelajari optimasi nonlinear adalah:

1. Memberikan **pemahaman teori dasar** yang kuat (gradien, Hessian, titik kritis, uji kelengkungan).
2. Melatih mahasiswa dalam **analisis matematis** menggunakan pendekatan manual maupun numerik.
3. Membekali mahasiswa dengan **keterampilan komputasi** agar mampu menyelesaikan permasalahan skala besar yang tidak mungkin dikerjakan hanya dengan tangan.
4. Membuka wawasan mengenai **aplikasi lintas disiplin**, sehingga mahasiswa mampu mengaitkan matematika dengan bidang ilmu lain.

**Fungsi yang Dikaji dalam Tugas Ini**

Dalam tugas ini, fungsi yang diteliti adalah:

Fungsi ini memiliki beberapa alasan penting untuk dipilih:

1. **Sifat periodik** → nilai fungsi berulang pada interval tertentu, sehingga terdapat banyak titik optimum.
2. **Kombinasi sinus dan cosinus** → menghadirkan keragaman bentuk grafik (naik, turun, puncak, lembah, dan pelana).
3. **Contoh ideal** → fungsi sederhana tetapi kaya makna, sehingga cocok untuk melatih analisis gradien, Hessian, serta metode Lagrange.

Dengan fungsi ini, mahasiswa dapat mempelajari bagaimana teori optimasi nonlinear diterapkan dalam kasus nyata, sekaligus melatih keterampilan interpretasi grafis melalui visualisasi kontur (2D) dan surface plot (3D).

1. Sejarah Singkat Optimasi Nonliniear

**Latar awal: kalkulus variasi dan kelahiran gagasan optimasi**

* Akar ide optimasi berasal dari **kalkulus variasi**, bidang yang berkembang pada abad ke-17 dan 18 untuk mencari fungsi yang meminimalkan atau memaksimalkan suatu fungsials (mis. lintasan terpendek, masalah brachistochrone). Tokoh-tokoh awal seperti **Jakob Bernoulli**, **Leonhard Euler**, dan **Joseph-Louis Lagrange** meletakkan dasar-dasar teori variasi yang kemudian menjadi landasan matematis optimasi.
* **Inti gagasan**: bukan sekadar mencari nilai ekstrem dari fungsi satu variabel, tetapi mencari fungsi atau kurva yang membuat nilai suatu integral ekstrem — ini memperluas konsep optimasi menjadi masalah tak hingga dimensi.

**Abad ke-18: Lagrange dan multiplier**

* **Joseph-Louis Lagrange** (1736–1813) memperkenalkan metode **Lagrange multiplier** dalam konteks mekanika analitik dan kalkulus variasi. Dalam bentuknya, metode ini memberikan cara sistematis untuk menemukan ekstrem fungsi f(x) dengan syarat kendala g(x)=0 melalui fungsi Lagrangian:
* **Signifikansi**: gagasan Lagrange mengubah cara berpikir tentang kendala — kendala tidak lagi “dihilangkan” dengan substitusi saja, tetapi diinkorporasi ke dalam kondisi stasioner yang seragam. Metode ini menjadi titik tolak untuk teori kendala modern (termasuk KKT).

**Abad ke-19: perkembangan konsep turunan, metode steepest descent**

* Pada abad ke-19 muncul pengembangan metode numerik awal untuk mencari solusi masalah ekstrem. Contoh: konsep **steepest descent** (sering dikaitkan dengan karya-karya awal seperti Cauchy) — yaitu bergerak menuju arah negatif gradien untuk menurunkan nilai fungsi.
* **Makna historis**: meskipun metode ini sederhana dan lambat untuk beberapa kasus, ia merupakan ide dasar yang berkembang menjadi metode optimasi numerik modern.

**Abad ke-20: teori kondisi orde pertama/ke-2, KKT, dan metode numerik modern**

* **KKT (Karush–Kuhn–Tucker)**: kondisi KKT menggeneralisasi gagasan Lagrange untuk kasus kendala persamaan dan pertidaksamaan. Konsep ini mengikat teori dualitas, multipliers, dan kondisi optimalitas dalam kerangka yang formal. (Catatan sejarah: formulasi KKT dipopulerkan oleh Kuhn & Tucker, walau gagasan awal dicatat sebelumnya oleh Karush.)
* **Newton untuk optimasi**: ekstensi Newton–Raphson untuk optimasi (memakai Hessian) memberi metode kuadratik cepat bila kondisi baik (konvergensi kuadratik dekat solusi).
* **Conjugate Gradient (CG)**: diperkenalkan oleh Hestenes & Stiefel (1952) sebagai metode efisien untuk menyelesaikan sistem linear besar dan kemudian diadaptasi untuk optimasi fungsi kuadrat besar dimensi — penting untuk masalah skala besar sebelum era komputer modern.
* **Quasi-Newton dan keluarga BFGS/DFP**: metode-metode ini (perkembangan dari akhir 1950-an sampai 1970an) memberikan aproksimasi invers Hessian sehingga menggabungkan kecepatan Newton dengan biaya komputasi yang lebih rendah; BFGS menjadi salah satu metode paling populer untuk optimasi tak-terbatas berdimensi menengah.
* **Dualitas & teori konveks**: di paruh kedua abad ke-20 teori konveks formal (mis. karya Rockafellar) memperkuat landasan teoretis optimasi konveks dan dualitas — memungkinkan hasil kuat tentang ketika solusi dual = solusi primal (strong duality).

**Era modern (akhir abad ke-20 — abad ke-21): komputasi, solver, dan penerapan skala besar**

* **Interior-point methods** (mis. Karmarkar, 1984 untuk LP) membuka jalan bagi algoritma yang efisien untuk masalah skala besar; metode-metode ini kemudian dikembangkan untuk masalah nonlinear dan convex besar.
* **Algoritma berbasis gradien stochastik**: munculnya data besar dan machine learning mendorong penggunaan **Stochastic Gradient Descent (SGD)** dan varian-variannya (Momentum, AdaGrad, RMSprop, Adam). Teknik-teknik ini khususnya penting untuk optimasi nonkonveks (mis. training deep neural networks) karena dapat menangani jumlah parameter yang sangat besar.
* **Solver & perangkat lunak**: berkembangnya paket-paket optimasi (mis. optimizers di MATLAB, SciPy, IPOPT, KNITRO, CVX) menjadikan metode teoritis dapat dipraktikkan pada kasus nyata dengan ukuran besar.
* **Metode hibrida & skala besar**: gabungan teknik (SQP, augmented Lagrangian, trust-region) dikembangkan untuk masalah nonlinier berkendala yang kompleks; teknik distribusi dan paralelisasi juga menjadi penting untuk masalah skala industri.

**Dampak interdisipliner & riset kontemporer**

* Optimasi nonlinear **menjadi pondasi riset interdisipliner** karena:
  + Banyak masalah nyata (rekayasa, ekonomi, bioinformatika, ML) diformulasikan sebagai masalah optimasi nonlinear.
  + Metode optimasi memicu perkembangan teori baru (mis. teori konveks, relaxasi semidefinite, optimasi global).
  + Tantangan modern meliputi: *non-konveksitas besar*, *skala dimensi tinggi*, *ketidakpastian data*, dan *kebutuhan komputasi paralel/distribusi*.
* **Arah riset saat ini**: global optimization (strategi untuk melampaui local minima), optimasi non-convex teoritis (sifat lanskap loss), metode acak dan adaptif untuk skala besar, serta integrasi optimasi dengan pembelajaran (meta-learning, hyperparameter optimization).

**Implikasi untuk pengajaran di Prodi Matematika**

* Sejarah ini relevan bukan sekadar trivia:
  + Menunjukkan mengapa metode klasik (Lagrange, KKT) tetap fundamental.
  + Menjelaskan asal-usul berbagai algoritma numerik yang diajarkan (mengapa BFGS, Newton, CG, dan SGD punya tempat masing-masing).
  + Mengaitkan kurikulum (teori + numerik + coding) dengan perkembangan nyata di lapangan, sehingga mahasiswa memahami *mengapa* mereka mempelajari topik tertentu, bukan hanya *bagaimana* menghitungnya.

1. Teori Dasar Optimasi Nonlinear — Titik Optimum dan Gradien

**Titik Optimum dan Gradien**

**Titik optimum** dibedakan menjadi:

* **Minimum global**: nilai terkecil pada seluruh domain.
* **Maksimum global**: nilai terbesar pada seluruh domain.
* **Minimum lokal**: nilai terkecil dibanding titik di sekitarnya.
* **Maksimum lokal**: nilai terbesar dibanding titik di sekitarnya.

Untuk menemukan titik optimum, digunakan **gradien**:

Syarat perlu agar f(x,y) mencapai ekstremum lokal adalah ∇f(x,y)=0. Setelah itu, sifat titik (minimum, maksimum, atau saddle) ditentukan dengan **Hessian** melalui uji tanda determinan dan turunan keduanya.

**Hessian dan Klasifikasi Titik Kritis**

Untuk mengklasifikasikan titik kritis digunakan **Hessian**, yaitu matriks turunan kedua:

Dengan kriteria:

* Jika det dan → **minimum lokal**.
* Jika det dan → **maksimum lokal**.
* Jika det → **saddle point**.
* Jika det → sifat titik tidak dapat ditentukan.

**Metode Lagrange**

Jika optimasi memiliki **kendala**, digunakan metode Lagrange.

Misalkan fungsi tujuan f(x,y) dengan kendala g(x,y)=0, maka dibentuk fungsi:

Syarat perlu optimum:

Artinya, turunan parsial terhadap x, y, dan λ semuanya sama dengan nol.  
Metode ini memungkinkan kita menemukan titik optimum tanpa harus menghilangkan kendala terlebih dahulu.

**Contoh Fungsi Sederhana**

Misalkan:

* Gradien:
* Titik Kritis: (0,0)
* Hessian:

Det dan , sehingga (0,0) Adalah minimum global.

1. Fungsi Utama

**Rentang nilai (mengapa maksimum = 2 dan minimum = −2)**

Karena dan untuk semua t, maka

Nilai maksimum 2 terwujud bila dan . Contoh titik yang memenuhi:

Nilai minimum terwujud bila dan . Contoh:

**Sifat periodik (banyak titik kritis)**

sin x dan cos y periodik dengan periode sehingga juga periodik dengan periode pada masing-masing koordinat. Semua fenomena ekstrem (maksimum, minimum, saddle) berulang pada kisi . Itu menjelaskan ada tak terhingga banyak titik kritis yang teratur.

**Gradien & Titik Kritis**

Turunan parsial (gradien):

Jadi gradien:

Titik Kritis (kondisi orde pertama)

Titik kritis diperoleh dari yaitu:

Solusi umum:

Karena persyaratan memisah per variabel, himpunan titik kritis merupakan himpunan produk dari titik-titik kritis untuk masing-masing fungsi satu-variabel.

Contoh numerik (beberapa titik kritis):

(maksimum), (minimum), (saddle).

**Hessian**

Turunan kedua:

Sehingga Hessian adalah matriks diagonal:

Karena , kelengkungan di arah x dan y terpisah (fungsi *separable*). Ini menyederhanakan analisis orde kedua: kita hanya perlu melihat tanda dan

**Uji Kritis**

Determinant Hessian:

Evaluasi pada titik kritis

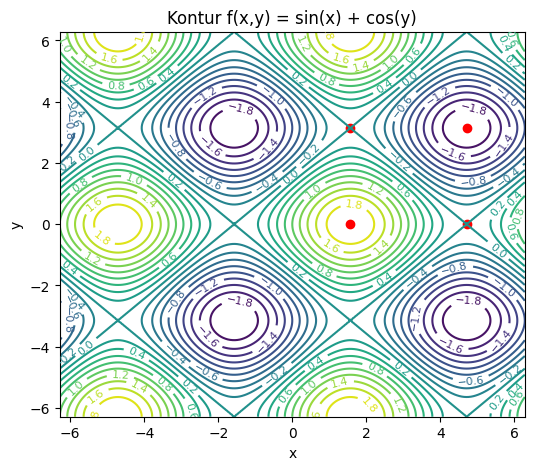
Maka

Klasifikasi berdasarkan paritas k,m (mudah diingat sebagai tabel):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| k mod 2 | m mod 2 | det(H) | f\_xx | Klasifikasi |
| 0 | 0 | +1 | -1 | Maksimum lokal (contoh: ) |
| 0 | 1 | -1 | -1 | Saddle point |
| 1 | 0 | -1 | +1 | Saddle point |
| 1 | 1 | +1 | +1 | Minimum lokal (contoh: ) |

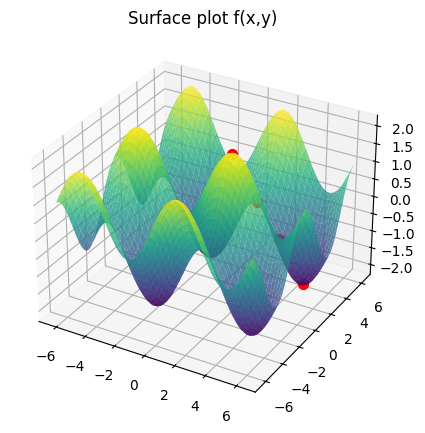
**Visualisasi**

* + 1. Kontur 2D pada rentang untuk x,y. Tandai titik kritis dengan simbol berbeda.



Interprestasi: daerah dengan garis rapat = perubahan tajam; pusat-pusat pola menunjukkan posisi puncak/lembah.

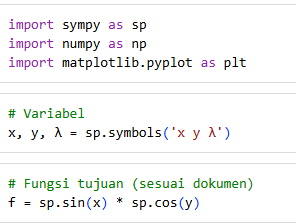
* + 1. Surface 3D (permukaan) yang memperlihatkan deretan puncak dan lembah.



3. Kontur dengan garis kendala, gambar garis memotong kontur.

# 

1. Fungsi Tujuan Tanpa Kendala Narasi Fungsi z=sin(x)+cos(y) memiliki pola periodik dengan puncak maksimum dan minimum yang berulang. Titik kritis terjadi saat gradien nol. Dengan menggunakan Hessian, dapat ditentukan apakah titik tersebut maksimum, minimum, atau saddle point. Hal ini menunjukkan karakteristik fungsi gelombang dua variabel yang memiliki banyak titik ekstrem.



Berikut adalah penjelasan langkah-langkah yang dilakukan dalam kode di atas, dilengkapi dengan formula matematikanya:

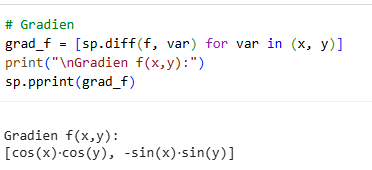
1. Gradien (Gradient):
   * Gradien dari suatu fungsi skalar multivariabel adalah vektor yang berisi turunan parsial pertama dari fungsi tersebut terhadap setiap variabelnya.
   * Secara matematis, gradien dari dilambangkan dengan (nabla ) dan didefinisikan sebagai:
   * Untuk fungsi kita, , gradiennya adalah:
   * Gradien menunjukkan arah peningkatan terbesar dari fungsi pada suatu titik.
2. Titik Kritis (Critical Points):
   * Titik kritis dari fungsi adalah titik-titik di mana gradien fungsi adalah vektor nol, atau di mana gradien tidak terdefinisi.
   * Secara matematis, titik kritis ditemukan dengan menyelesaikan sistem persamaan:
   * Artinya, setiap komponen gradien harus bernilai nol:
   * Untuk fungsi kita, titik kritis ditemukan dengan menyelesaikan:
   * Pada titik kritis, fungsi dapat mencapai nilai maksimum lokal, minimum lokal, atau saddle point.
3. Hessian (Hessian Matrix):
   * Matriks Hessian dari fungsi skalar adalah matriks persegi dari turunan parsial kedua dari fungsi tersebut.
   * Elemen pada baris dan kolom dari matriks Hessian, dilambangkan , adalah turunan parsial kedua dari terhadap dan :
   * Matriks Hessian secara umum ditulis sebagai:
   * Untuk fungsi kita, , matriks Hessiaannya adalah:
   * Matriks Hessian digunakan untuk menguji sifat titik kritis (apakah itu maksimum, minimum, atau saddle point) berdasarkan kriteria turunan kedua. Analisis ini melibatkan determinan dan/atau eigenvalues dari matriks Hessian yang dievaluasi pada titik kritis.

Langkah selanjutnya dalam kode adalah menganalisis matriks Hessian di setiap titik kritis untuk menentukan jenis titik kritis tersebut.

Berikut adalah penjelasan langkah-langkah yang dilakukan dalam kode di atas:

1. Gradien (Gradient):
   * Gradien dari suatu fungsi multivariabel adalah vektor yang berisi turunan parsial pertama dari fungsi tersebut terhadap setiap variabelnya.
   * Gradien menunjukkan arah peningkatan terbesar dari fungsi pada suatu titik.
   * Dalam kode, sp.diff(f, var) digunakan untuk menghitung turunan parsial fungsi f terhadap variabel x dan y. Hasilnya adalah grad\_f.
2. Titik Kritis (Critical Points):
   * Titik kritis adalah titik-titik di mana gradien fungsi adalah nol atau tidak terdefinisi.
   * Pada titik kritis, fungsi dapat mencapai nilai maksimum lokal, minimum lokal, atau saddle point.
   * Dalam kode, sp.solve(grad\_f, (x, y), dict=True) digunakan untuk mencari nilai x dan y di mana komponen-komponen gradien (grad\_f) bernilai nol. Hasilnya disimpan dalam list kritik.
3. Hessian (Hessian Matrix):
   * Matriks Hessian adalah matriks persegi dari turunan parsial kedua dari suatu fungsi.
   * Matriks Hessian digunakan untuk menguji sifat titik kritis (apakah itu maksimum, minimum, atau saddle point) berdasarkan kriteria turunan kedua.
   * Dalam kode, sp.hessian(f, (x, y)) digunakan untuk menghitung matriks Hessian dari fungsi f terhadap variabel x dan y. Hasilnya adalah matriks H.

Langkah selanjutnya dalam kode adalah menganalisis matriks Hessian di setiap titik kritis untuk menentukan jenis titik kritis tersebut (menggunakan determinan dan eigenvalues Matriks Hessian).



Titik kritis ditemukan dengan menyelesaikan sistem persamaan , yang berarti setiap turunan parsial pertama fungsi harus bernilai nol. Formula matematis selengkapnya dijelaskan pada bagian “Titik Kritis (Critical Points)” di sel Markdown di atas.

Untuk mendapatkan titik-titik kritis, kita perlu menyelesaikan sistem persamaan berikut:

Dari persamaan (1), , ini benar jika: \* , yang terjadi saat , di mana adalah bilangan bulat. \* , yang terjadi saat , di mana adalah bilangan bulat.

Dari persamaan (2), , ini benar jika: \* , yang terjadi saat , di mana adalah bilangan bulat. \* , yang terjadi saat , di mana adalah bilangan bulat.

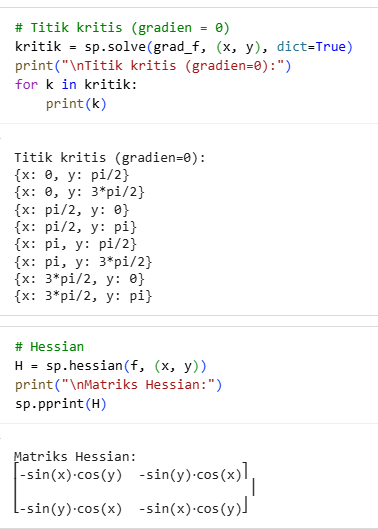
Untuk memenuhi *kedua* persamaan secara bersamaan, kita harus mencari nilai dan yang memenuhi kondisi dari persamaan (1) *dan* persamaan (2).

Mari kita analisis kemungkinannya:

* Kasus 1**:**  Ini berarti . Jika ini benar, maka untuk memenuhi persamaan (2), kita harus memiliki . Karena tidak pernah nol (nilainya 1 atau -1), maka haruslah . Ini terjadi saat . Jadi, satu set solusi adalah dan .
* Kasus 2**:**  Ini berarti . Jika ini benar, maka untuk memenuhi persamaan (2), kita harus memiliki . Karena tidak pernah nol (nilainya 1 atau -1), maka haruslah . Ini terjadi saat . Jadi, set solusi lainnya adalah dan .

Menggabungkan kedua kasus ini, titik-titik kritis terjadi ketika salah satu dari kondisi berikut terpenuhi: 1. dan (di mana adalah bilangan bulat) 2. dan (di mana adalah bilangan bulat)

Titik-titik kritis yang ditampilkan di output kode adalah beberapa contoh dari solusi umum ini dalam rentang tertentu.



Untuk mendapatkan nilai-nilai eigen (eigenvalues) dari matriks Hessian yang dievaluasi pada suatu titik kritis, kita perlu menyelesaikan persamaan karakteristik:

di mana adalah matriks Hessian yang dievaluasi pada titik kritis, adalah matriks identitas berukuran sama dengan , dan adalah nilai eigen yang ingin kita cari.

Untuk matriks Hessian 2x2 seperti kasus kita, , persamaan karakteristiknya menjadi:

Ini adalah persamaan kuadrat dalam . Solusi dari persamaan kuadrat ini akan memberikan nilai-nilai eigen dan .

Sebagai contoh, untuk titik kritis , matriks Hessian yang dievaluasi adalah . Maka . Persamaan karakteristiknya adalah:

Menyelesaikan persamaan ini:

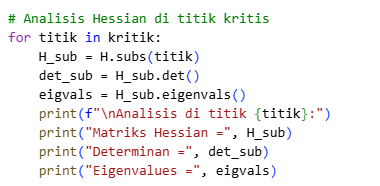
Jadi, nilai eigen untuk titik adalah dan . Ini sesuai dengan output kode {-1: 1, 1: 1} yang menunjukkan nilai eigen -1 dengan multiplicitas 1 dan nilai eigen 1 dengan multiplicitas 1.

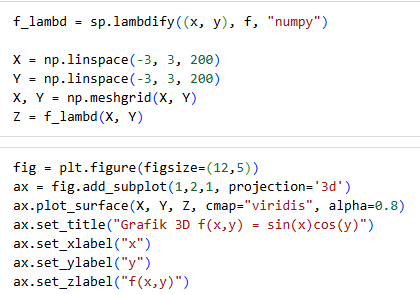
Contoh lain, untuk titik kritis , matriks Hessian yang dievaluasi adalah . Maka . Persamaan karakteristiknya adalah:

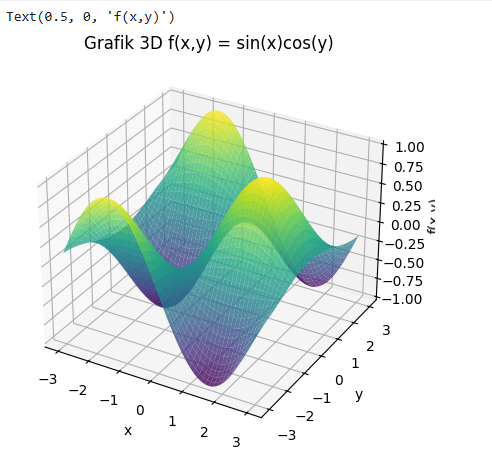
Menyelesaikan persamaan ini:

Jadi, nilai eigen untuk titik adalah dan (nilai eigen ganda). Ini sesuai dengan output kode {-1: 2} yang menunjukkan nilai eigen -1 dengan multiplicitas 2.

Analisis yang sama dilakukan untuk semua titik kritis lainnya seperti yang ditunjukkan pada output kode. Tanda dari nilai-nilai eigen ini (atau determinan dan elemen diagonal pertama matriks Hessian) kemudian digunakan untuk mengklasifikasikan titik kritis sebagai maksimum lokal, minimum lokal, atau titik pelana, seperti yang dijelaskan pada sel Markdown sebelumnya.

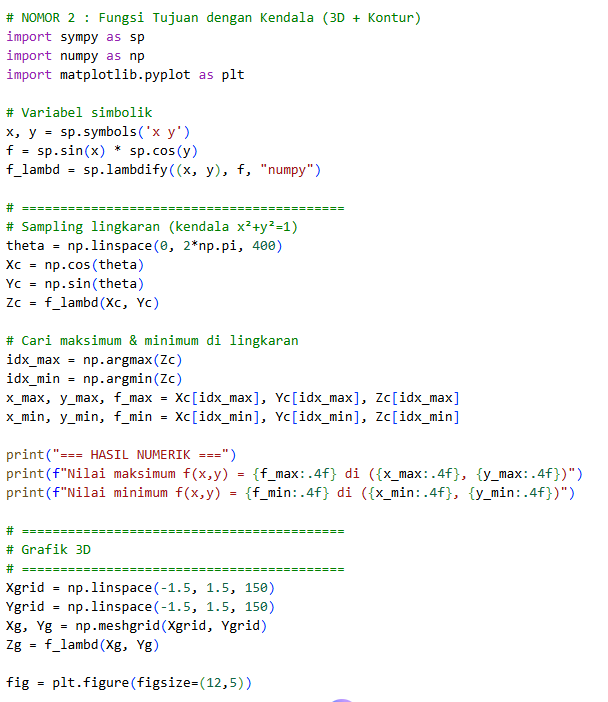


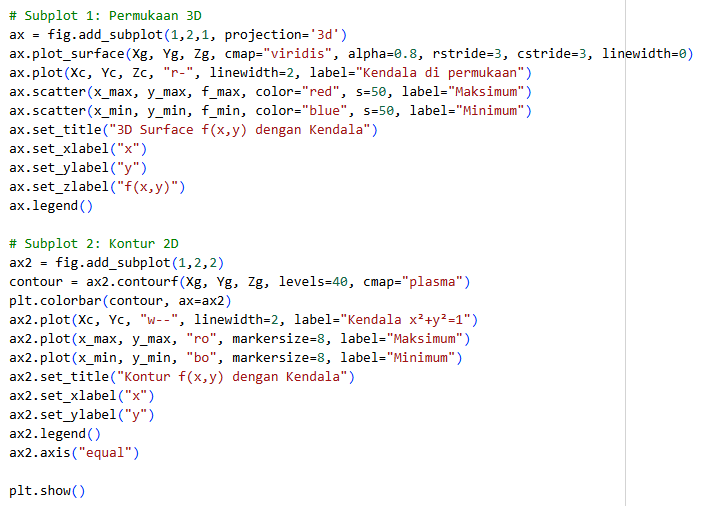


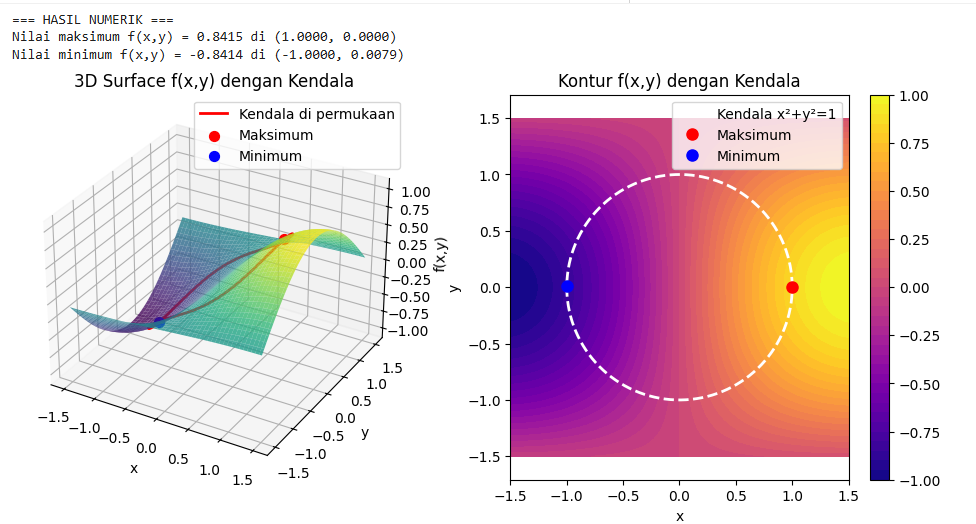


1. Fungsi Tujuan dengan kendala

Dengan adanya kendala, pencarian maksimum/minimum menjadi terbatas hanya pada domain tertentu. Konsep gradien fungsi dan kendala harus searah pada titik optimum (prinsip gradien sebanding). Hal ini menunjukkan bagaimana optimisasi dengan kendala mengubah lokasi titik ekstrim yang mungkin berbeda dengan fungsi tanpa kendala.





****

# Glosarium Fungsi Tujuan Tanpa Kendala

### 1. Fungsi Tujuan (Objective Function)

* Definisi: Fungsi matematika yang nilainya ingin dikecilkan (minimasi) atau dibesarkan (maksimasi).
* Analogi: Seperti target skor yang ingin dicapai dalam game.
* Ilustrasi:
  + → kita cari titik terendah dari kurva parabola.

### 2. Tanpa Kendala (Unconstrained)

* Definisi: Optimasi dilakukan tanpa syarat tambahan. Semua nilai variabel boleh dicoba.
* Analogi: Seperti mencari tempat duduk paling nyaman di ruangan tanpa aturan “hanya boleh di barisan depan/belakang”.
* Ilustrasi: Kurva parabola , kita bisa pilih bebas tanpa batasan.

### 3. Optimasi (Optimization)

* Definisi: Proses mencari nilai terbaik dari fungsi (minimum atau maksimum).
* Analogi: Seperti mencari harga barang termurah di toko online atau skor tertinggi di game.
* Ilustrasi:
  + Minimum → “harga termurah”
  + Maksimum → “skor tertinggi”

### 4. Minimum Global vs Minimum Lokal

* Definisi:
  + *Global:* titik terendah di seluruh grafik.
  + *Lokal:* titik terendah di sekitar lingkungannya saja.
* Analogi:
  + *Global:* gunung tertinggi di dunia = Gunung Everest.
  + *Lokal:* gunung tertinggi di Jawa = Semeru (tapi bukan yang tertinggi di dunia).
* Ilustrasi: Fungsi bergelombang (seperti ombak), tiap lembah = minimum lokal, lembah terdalam = minimum global.

### 5. Maksimum Global vs Maksimum Lokal

* Sama persis dengan minimum, tapi untuk titik tertinggi.

### 6. Saddle Point

* Definisi: Titik stasioner yang bukan minimum atau maksimum. Fungsi bisa naik di satu arah, turun di arah lain.
* Analogi: Bentuk pelana kuda, ada sisi naik dan sisi turun.
* Ilustrasi: .

### 7. Gradien (Gradient)

* Definisi: Vektor turunan pertama, menunjukkan arah perubahan tercepat.
* Analogi: Seperti kompas yang selalu menunjuk ke arah paling curam di gunung.
* Ilustrasi: Jika di gunung → gradien menunjukkan arah tercepat menanjak.

### 8. Hessian

* Definisi: Matriks turunan kedua, dipakai untuk menentukan sifat titik stasioner (minimum/maksimum/saddle).
* Analogi: Seperti memeriksa bentuk mangkok:
  + Mangkok → minimum.
  + Kubah → maksimum.
  + Pelana → saddle.

### 9. Titik Stasioner (Stationary Point / Critical Point)

* Definisi: Titik di mana gradien = 0. Bisa minimum, maksimum, atau saddle.
* Analogi: Di gunung → tempat di mana jalan rata (tidak menanjak/menurun).
* Ilustrasi: , di gradiennya nol → titik minimum.

### 10. Convex vs Non-Convex

* Definisi:
  + Convex: fungsi berbentuk mangkok, hanya ada satu minimum global.
  + Non-convex: fungsi bergelombang, bisa banyak minimum lokal.
* Analogi:
  + Convex = mangkok → semua bola jatuh ke dasar yang sama.
  + Non-convex = gunung + lembah → bola bisa nyangkut di lembah lokal.

# Glosarium Fungsi Tujuan dengan Kendala (Constrained Optimization)

### 1. Kendala (Constraint)

* Definisi sederhana: Aturan atau batasan yang harus dipenuhi dalam mencari solusi.
* Detail teori: Kendala bisa berupa persamaan (*equality*) atau pertidaksamaan (*inequality*).
  + *Equality constraint:*
  + *Inequality constraint:*
* Analogi sehari-hari: Kamu ingin membeli barang dengan harga termurah, tapi kendalanya adalah uang di dompetmu terbatas.
* Contoh matematis:

### 2. Daerah Feasible (Feasible Region)

* Definisi sederhana: Semua titik yang memenuhi kendala.
* Detail teori: Optimasi hanya boleh dilakukan di dalam wilayah ini, bukan di seluruh ruang.
* Analogi sehari-hari: Kalau kamu main bola di lapangan, daerah feasible = area dalam garis putih. Bola keluar lapangan → tidak feasible.
* Contoh matematis: Kendala → daerah feasible berbentuk segitiga di kuadran pertama.

### 3. Fungsi Tujuan dengan Kendala (Constrained Objective Function)

* Definisi sederhana: Fungsi tujuan yang nilainya dicari hanya di dalam daerah feasible.
* Detail teori: Kadang titik optimum fungsi tujuan tanpa kendala tidak valid, karena melanggar syarat.
* Analogi sehari-hari: Kamu ingin tidur selama mungkin (maksimasi waktu tidur), tapi ada kendala: besok kuliah jam 7 pagi. Jadi optimum waktumu dibatasi oleh kendala itu.
* Contoh: dengan kendala . Optimum bukan di tak hingga, tapi di .

### 4. Lagrangian Function

* Definisi sederhana: Fungsi baru yang menggabungkan fungsi tujuan dan kendala.
* Detail teori:
* Digunakan untuk menyelesaikan masalah optimasi dengan persamaan kendala.
* Analogi sehari-hari: Seperti gabungan resep masakan + aturan diet. Kamu tetap makan enak (fungsi tujuan), tapi ada tambahan larangan gula (kendala).
* Contoh: , kendala .

### 5. Multiplier Lagrange (λ)

* Definisi sederhana: Variabel tambahan yang “memaksa” kendala dipatuhi.
* Detail teori: λ menunjukkan seberapa besar perubahan fungsi tujuan jika kendala dilonggarkan sedikit.
* Analogi sehari-hari: Seperti “harga bayangan”: kalau dompetmu ditambah Rp 1.000, seberapa besar tambahan kenyamanan yang kamu dapat?
* Contoh: Jika λ=2, artinya setiap kenaikan 1 unit pada batas kendala akan meningkatkan fungsi tujuan sebesar 2.

### 6. KKT Conditions (Karush–Kuhn–Tucker)

* Definisi sederhana: Aturan matematis untuk menentukan optimum dengan kendala pertidaksamaan.
* Detail teori: KKT menggabungkan:
  + Turunan pertama (gradien) = 0
  + Kendala harus dipenuhi
  + λ≥0 untuk pertidaksamaan
  + *Complementary slackness* (kendala aktif → λ>0, kendala tidak aktif → λ=0)
* Analogi sehari-hari: Seperti peraturan lalu lintas: bukan hanya jalan lurus, tapi juga lampu merah, marka jalan, dan rambu batas kecepatan. Semua harus dicek.
* Contoh: . Solusi KKT memberi optimum di , bukan di .

### 7. Active Constraint

* Definisi sederhana: Kendala yang berlaku sebagai persamaan pada titik optimum.
* Detail teori: Jika dan di optimum , maka kendala aktif.
* Analogi sehari-hari: Kamu ingin tidur 8 jam, tapi karena ada kuliah jam 7, kamu cuma bisa tidur 6 jam. Kendala kuliah aktif.
* Contoh: Kendala , optimum di titik . Karena , maka kendala aktif.

### 8. Shadow Price (Harga Bayangan)

* Definisi sederhana: Nilai tambahan fungsi tujuan jika kendala dilonggarkan sedikit.
* Detail teori: Sama dengan nilai multiplier Lagrange.
* Analogi sehari-hari: Jika tambahan Rp 1.000 di dompetmu bisa menambah kenikmatan makan sebesar 10 poin → shadow price = 10.
* Contoh: Dalam program linear, shadow price = kenaikan keuntungan jika kapasitas produksi bertambah 1 unit.

### 9. Binding vs Non-Binding Constraint

* Definisi sederhana:
  + *Binding:* benar-benar membatasi solusi.
  + *Non-binding:* tidak berpengaruh ke solusi.
* Detail teori: Binding = kendala aktif. Non-binding = kendala masih longgar di optimum.
* Analogi sehari-hari:
  + Binding: uang pas-pasan, jadi benar-benar membatasi belanja.
  + Non-binding: uangmu banyak, jadi batas dompet tidak relevan.
* Contoh: Optimasi dengan kendala . Jika solusi optimum , maka binding. Jika optimum , maka non-binding