Matgeo Presentation - Problem 3.2.4

ee25btech11063 - Vejith

September 1, 2025

Question

Construct the triangle BD′C′ similar to △BDC with scale factor $\frac{4}{3}$.Draw the line segment D′A′. parallel to DA where A^p prime lies on extended side BA.Is A′BC′D′ a parallelogram?

Description

Solution:

Point	Name
$\begin{pmatrix} 0 \\ 0 \end{pmatrix}$	Point A
$\begin{pmatrix} 4 \\ 0 \end{pmatrix}$	Point B
$\begin{pmatrix} 4 \\ 3 \end{pmatrix}$	Point C
$\begin{pmatrix} 0 \\ 3 \end{pmatrix}$	Point D
$\begin{pmatrix} -4/3 \\ 4 \end{pmatrix}$	Point D′
$\begin{pmatrix} 4 \\ 4 \end{pmatrix}$	Point C′
$\begin{pmatrix} -4/3 \\ 0 \end{pmatrix}$	Point A′

Table: Variables Used

Solution

consider $\triangle$BDC.constructs a $\triangle$BD$'$C$'$ with scale factor $\frac{4}{3}$.
This means

$$\triangle BD'C' \sim \triangle BDC. \tag{0.1}$$

$$\frac{BD'}{BD} = \frac{BC'}{BC} = \frac{D'C'}{DC} = \frac{4}{3}. \tag{0.2}$$

So D$'$ lies on extension of BD and C$'$.

Construct A$'$

Draw D$'$A$'$ $\|$ DA with A$'$ on extension of BA.

Check the parallelogram property

1.By construction D$'$A$'$ $\|$ DA.
But since DA $\|$ C$'$B(by similarity of triangles),we get:

$$D'A' \parallel BC'. \tag{0.3}$$

2.A$'$ lies on extended BA,we have :

$$A'B \parallel D'C'. \tag{0.4}$$

Conclusion

Thus:

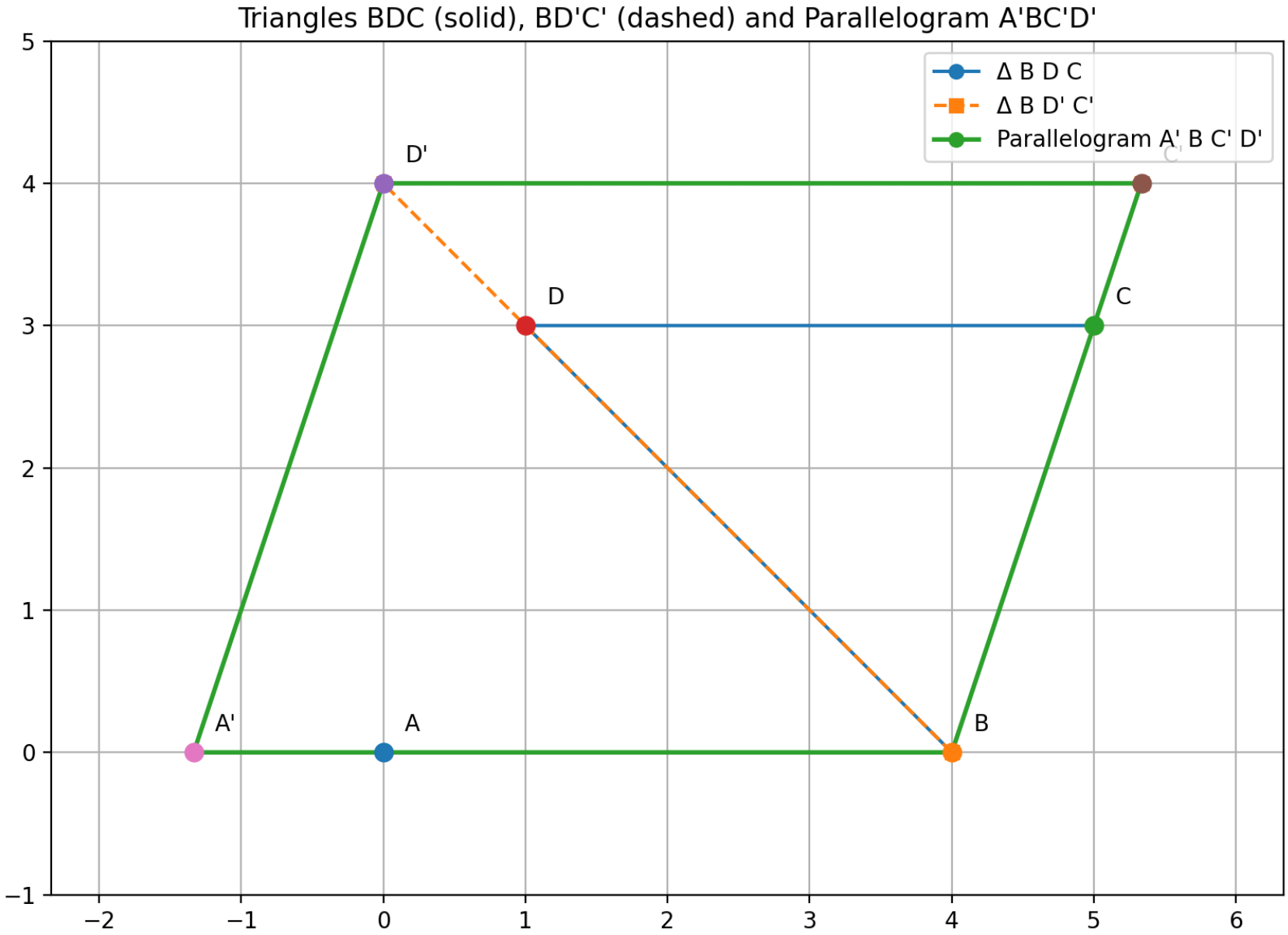
$$A'B \parallel D'C'. \tag{0.5}$$

$$D'A' \parallel BC'. \tag{0.6}$$

so,opposite sides are parallel.

$\implies$ A′BC′D′ is a parallelogram

Plot


Triangles BDC (solid), BD'C' (dashed) and Parallelogram A'BC'D'
Δ B D C
Δ B D' C'
Parallelogram A' B C' D'
D'
D
C
A'
A
B
5
4
3
2
1
0
−1
−2
−1
0
1
2
3
4
5
6

C Code: triangle.c

```c
/* triangle.c
   - writes points to "triangle.dat"
   - computes A' so that D'A' || DA and A' lies on extended BA
   - checks whether A' B C' D' is a parallelogram
*/
#include <stdio.h>
#include <math.h>

typedef struct { double x, y; } Point;

int areParallel(Point p1, Point p2, Point q1, Point q2) {
    double dx1 = p2.x - p1.x, dy1 = p2.y - p1.y;
    double dx2 = q2.x - q1.x, dy2 = q2.y - q1.y;
    return fabs(dy1 * dx2 - dy2 * dx1) < 1e-8;
}

int main(void) {
    FILE *fp = fopen("triangle.dat", "w");
    if (!fp) {
        perror("fopen");
        return 1;
    }

    /* Choose ABCD to be a parallelogram (so final shape will be a parallelogram).
       Example: rectangle/parallelogram with A=(0,0), B=(4,0), D=(0,3).
       Then C = B + D - A = (4,3).
    */
    Point A = {0.0, 0.0};
    Point B = {4.0, 0.0};
    Point D = {0.0, 3.0};
    Point C = { B.x + D.x - A.x, B.y + D.y - A.y }; /* ensures ABCD is parallelogram */

    double k = 4.0 / 3.0;
```

C Code: triangle.c

```
/* BD'C' similar to BDC with scale factor k: D' = B + k*(D - B), C' = B + k*(C - B) */
Point Dp = { B.x + k * (D.x - B.x), B.y + k * (D.y - B.y) };
Point Cp = { B.x + k * (C.x - B.x), B.y + k * (C.y - B.y) };

/* Solve for t where A' = B + t*(A - B) and D'A' || DA.
   Derivation:
     Let v = A - D.
     Let u(t) = (B - D') + t*(A - B). (u = A' - D')
     Parallel condition: u.x * v.y - u.y * v.x = 0
   => t = [ (B.y - D'.y)*v.x - (B.x - D'.x)*v.y ]
          / [ (A.x - B.x)*v.y - (A.y - B.y)*v.x ]
*/
double vx = A.x - D.x;
double vy = A.y - D.y;
double numerator = (B.y - Dp.y) * vx - (B.x - Dp.x) * vy;
double denominator = (A.x - B.x) * vy - (A.y - B.y) * vx;

if (fabs(denominator) < 1e-12) {
    fprintf(stderr, "Denominator ~ 0: can't find unique A' (degenerate configuration)\n");
    fclose(fp);
    return 1;
}

double t = numerator / denominator;
Point Ap = { B.x + t * (A.x - B.x), B.y + t * (A.y - B.y) };

/* Write coordinates */
fprintf(fp, "A   = (%.6f, %.6f)\n", A.x, A.y);
fprintf(fp, "B   = (%.6f, %.6f)\n", B.x, B.y);
fprintf(fp, "C   = (%.6f, %.6f)\n", C.x, C.y);
fprintf(fp, "D   = (%.6f, %.6f)\n", D.x, D.y);
fprintf(fp, "D' = (%.6f, %.6f)\n", Dp.x, Dp.y);
fprintf(fp, "C' = (%.6f, %.6f)\n", Cp.x, Cp.y);
```

C Code: triangle.c

```
    fprintf(fp, "A' = (%.6f, %.6f)\n", Ap.x, Ap.y);

    /* Check parallelogram: opposite sides parallel */
    int cond1 = areParallel(Ap, B, Dp, Cp); /* A'B || D'C' */
    int cond2 = areParallel(Ap, Dp, B, Cp); /* A'D' || B C' */

    if (cond1 && cond2) {
        fprintf(fp, "\nA'BC'D' is a parallelogram.\n");
        printf("A'BC'D' is a parallelogram.\n");
    } else {
        fprintf(fp, "\nA'BC'D' is NOT a parallelogram.\n");
        printf("A'BC'D' is NOT a parallelogram.\n");
    }

    fclose(fp);
    return 0;
}
```

Python: plot.py

```
   import numpy as np
import matplotlib.pyplot as plt

# --- set up a non-rectangular parallelogram ABCD (so the plot won't look square)
A = np.array([0.0, 0.0])
B = np.array([4.0, 0.0])
D = np.array([1.0, 3.0]) # note: not (0,3)  this slants the shape
C = B + D - A # ensures ABCD is a parallelogram

k = 4.0 / 3.0 # scale factor for triangle BD'C' similar to BDC

# scaled triangle BD'C'
Dp = B + k * (D - B)
Cp = B + k * (C - B)

# Solve for t where A' = B + t*(A - B) and (A' - D') is parallel to (A - D)
v = A - D # vector DA (we use A - D so later we test parallelism with A' - D')
numerator = (B[1] - Dp[1]) * v[0] - (B[0] - Dp[0]) * v[1]
denominator = (A[0] - B[0]) * v[1] - (A[1] - B[1]) * v[0]
if abs(denominator) < 1e-12:
    raise RuntimeError("Degenerate configuration: can't compute A' (denominator ~ 0).")
t = numerator / denominator
Ap = B + t * (A - B)

# small helper for 2D cross product (scalar)
def cross2(u, v):
    return u[0]*v[1] - u[1]*v[0]

# Check parallelogram: opposite sides parallel
vec_ApB = B - Ap
vec_DpCp = Cp - Dp
vec_ApDp = Dp - Ap
vec_BCp = Cp - B
```

Python: plot.py

```
cond1 = abs(cross2(vec_ApB, vec_DpCp)) < 1e-8
cond2 = abs(cross2(vec_ApDp, vec_BCp)) < 1e-8
is_parallelogram = cond1 and cond2

print("Points:")
for name, p in [("A",A),("B",B),("C",C),("D",D),("D'",Dp),("C'",Cp),("A'",Ap)]:
    print(f"{name:3} = ({p[0]:.6f}, {p[1]:.6f})")
print("\nChecks:")
print(" A'B  D'C' :", cond1)
print(" A'D'  BC' :", cond2)
print("=> A'BC'D' is parallelogram?:", is_parallelogram)

# --- Plotting ---
plt.figure(figsize=(9,6))

# helper to plot and close polygons
def plot_poly(pts, style, label, z=1, lw=1.5):
    pts_closed = np.vstack([pts, pts[0]])
    plt.plot(pts_closed[:,0], pts_closed[:,1], style, label=label, linewidth=lw, zorder=z)

# triangles and parallelogram
plot_poly(np.array([B, D, C]), '-o', " BDC", z=2, lw=1.5)
plot_poly(np.array([B, Dp, Cp]), '--s', " BD'C'", z=2, lw=1.5)
plot_poly(np.array([Ap, B, Cp, Dp]), '-o', "Parallelogram A'BC'D'", z=3, lw=2)

# label points with offsets
pts = {"A":A,"B":B,"C":C,"D":D,"D'":Dp,"C'":Cp,"A'":Ap}
for name, p in pts.items():
    plt.scatter(p[0], p[1], s=60, zorder=5)
    plt.text(p[0] + 0.15, p[1] + 0.15, name, fontsize=10)

# set axis limits with margins so shape is clear
all_pts = np.vstack(list(pts.values()))
```

Python: plot.py

```
xmin, ymin = all_pts.min(axis=0) - 1.0
xmax, ymax = all_pts.max(axis=0) + 1.0
plt.xlim(xmin, xmax)
plt.ylim(ymin, ymax)

plt.gca().set_aspect('equal', adjustable='box')
plt.grid(True)
plt.legend()
plt.title("Triangles␣BDC␣(solid),␣BD'C'␣(dashed)␣and␣Parallelogram␣A'BC'D'")
plt.tight_layout()

# save and show
plt.savefig("parallelogram_plot.png", dpi=200)
plt.show()
```