

Éléments de correction du TP1

```
> MySolve := proc (f, x, a, b)
  RETURN ( { solve( {f, x ≥ a, x ≤ b}, x) } );
end:
> MySolve(x2 - 1, x, 0, 2);
      {{x= 1}}
```

(1)

```
> MySolve(x3 - 0.25·x2 - 0.875·x + 0.375, x, 0, 1);
      {{x= 0.5000000000}, {x= 0.7500000000}}
```

(2)

```
> MySolve(x3 - 0.25·x2 - 0.875·x + 0.375, x, -2, 2);
      {{x= -1.}, {x= 0.5000000000}, {x= 0.7500000000}}
```

(3)

```
> MyNbRealRoots := proc (f, x, a, b)
  RETURN ( nops ( MySolve(f, x, a, b) ) );
end:
> MyNbRealRoots (x2 - 1, x, 0, 2);
      1
```

(4)

```
> MyNbRealRoots (x3 - 0.25·x2 - 0.875·x + 0.375, x, 0, 1);
      2
```

(5)

```
> MyNbRealRoots (x3 - 0.25·x2 - 0.875·x + 0.375, x, -2, 2);
      3
```

(6)

```
> MyNbRealRoots (x3 - 0.25·x2 - 0.875·x + 0.375, x, 10, 20);
      0
```

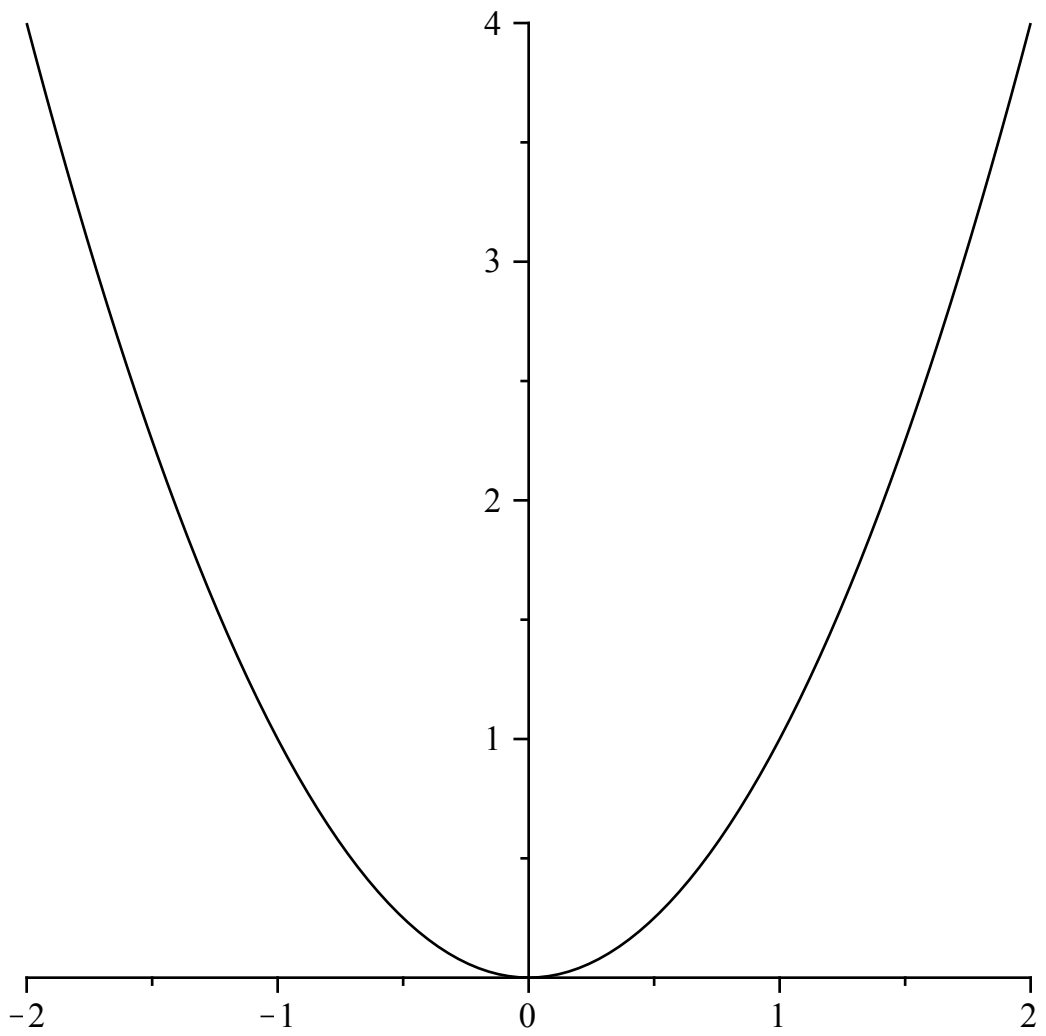
(7)

```
> with ( plots ) :
```

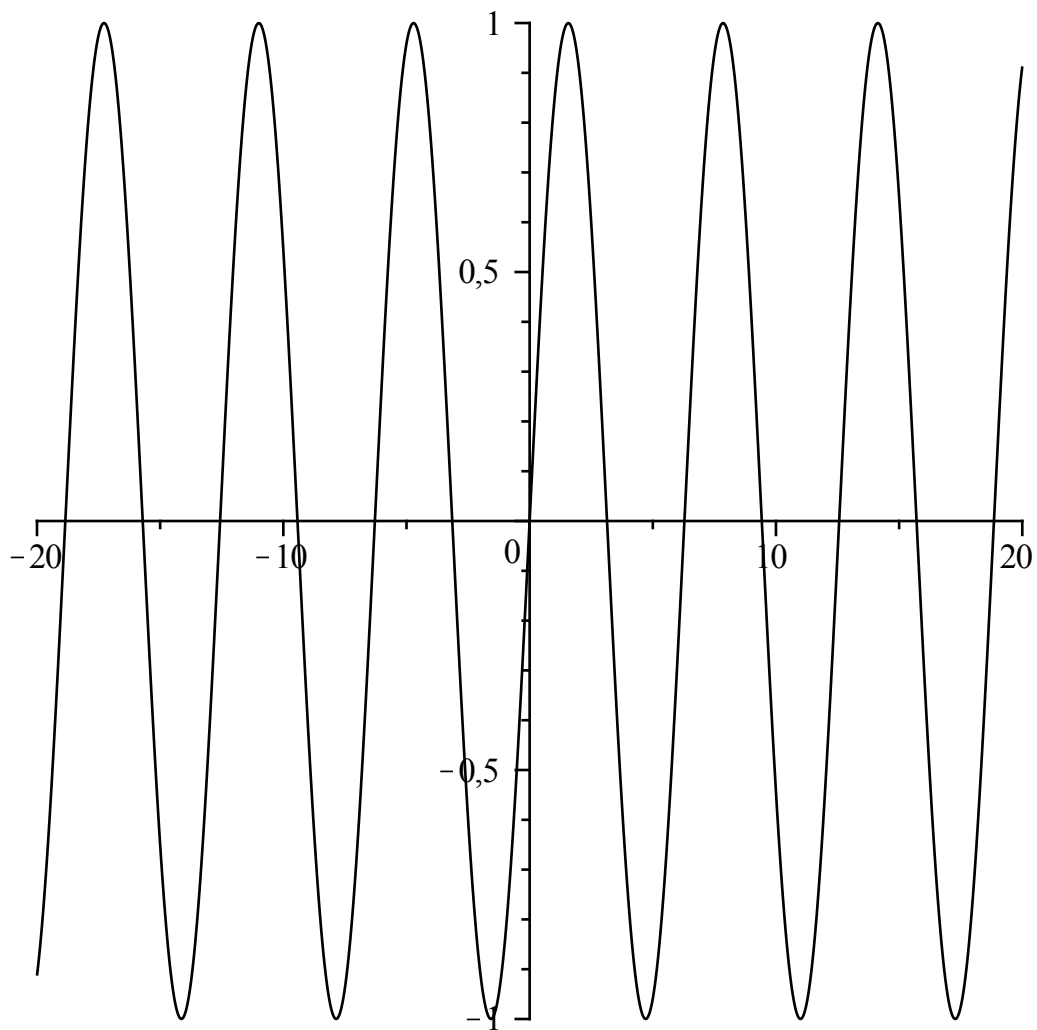
La fonction MyFuncPlot permet de tracer des graphes de fonctions.

```
> MyFuncPlot := proc (f, x, a, b, grid)
  RETURN ( CURVES ( [ seq ( [ a + (b-a)·i / grid, evalf ( eval ( f, x = a + (b-a)·i / grid ) ) ] ),
    i = 0 .. grid ] ) );
end:
```

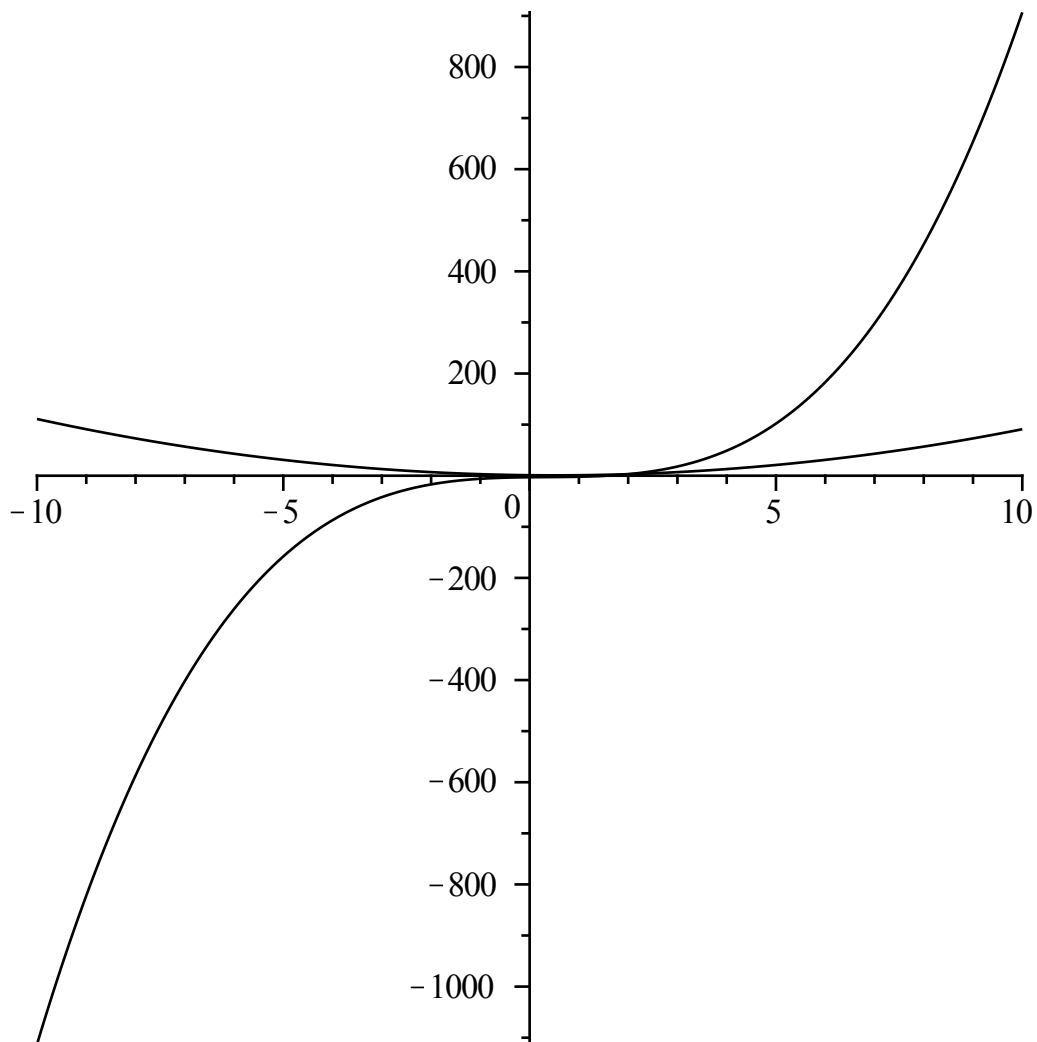
```
> P := MyFuncPlot(x2, x, -2, 2, 100) : display(P);
```



```
> P := MyFuncPlot(sin(x), x, -20, 20, 1000) : display(P);
```



```
> P := MyFuncPlot( $x^3 - x^2 + x - 3$ , x, -10, 10, 1000) : Q := MyFuncPlot( $x^2 - x + 1$ , x,  
-10, 10, 1000) : display(P, Q);
```



```

> MyNewtonMethod := proc ( f, x, z, prec )
  local err, znew, zold;
  err := prec + 1;
  zold := z;
  while ( err ≥ prec ) do
    znew := zold - evalf (  $\frac{\text{eval}(f, x = zold)}{\text{eval}(\text{diff}(f, x), x = zold)}$  );
    err := norm(zold - znew, 2);
    zold := znew;
  od;
  RETURN (zold);
end;

```

Ci-dessus la méthode de Newton.

```

> MyNewtonMethod (  $y^2 - 1$ , y, -9, 0.001 );
-1.000000000

```

(8)

```

> MyNewtonMethod (  $y^2 - 1$ , y, 1.2, 0.001 );
1.000000009

```

(9)

```

> MyNewtonMethod (  $y^2 - 1$ , y, -9, 0.0001 );

```

....

-1.000000000 (10)

Les fonctions pour récupérer les parties réelles et imaginaires des polynomes.

```
> RealPart := proc (f, x)
  local coef, d;
  d := degree(f, x);
  RETURN (convert([seq(ℜ(coeff(f, x, i)) · xi, i = 0 .. d)], '+' ));
end;
```

```
> f := (1 + 2·I) · x2 - 3 + 5·I;
      f := (1 + 2 I) x2 - 3 + 5 I (11)
```

```
> RealPart(f, x);
      -3 + x2 (12)
```

```
> ImagPart := proc (f, x)
  local coef, d;
  d := degree(f, x);
  RETURN (convert([seq(ℑ(coeff(f, x, i)) · xi, i = 0 .. d)], '+' ));
end;
```

```
> ImagPart(f, x);
      5 + 2 x2 (13)
```

Fonction permettant de tester si un polynome à coefficients complexes a des racines réelles.

```
> HasRealRoot := proc (f, x, a, b)
  local P;
  P := gcd(RealPart(f, x), ImagPart(f, x));
  if (MyNbRealRoots(P, x, a, b) > 0) then
    RETURN (1);
  else
    RETURN (0);
  end if;
end;
```

```
> HasRealRoot(f, x, 0, 1);
      0 (14)
```

```
> f := (1 + I) · x2 - (1 + I);
      f := (1 + I) x2 - 1 - I (15)
```

```
> HasRealRoot(f, x, 0, 1);
      1 (16)
```

Calcul d'un polynome sous forme développée à partir de ces racines.

```
> PolFromRoots := proc (Z, x)
  RETURN (expand(convert([seq(Z[i] - x, i = 1 .. nops(Z)], '*'))));
end;
```

```
> PolFromRoots([-1 - I, 1 + I], y);
      -2 I + y2 (17)
```

```
> Homotop := proc (Z, f, t, x)
  RETURN (t·f + (1 - t) · PolFromRoots(Z, x));
```

end:

```
> F := Homotop([-1, 1], f, t, x);  
      F := t((1 + I) x^2 - 1 - I) + (1 - t) (-1 + x^2) (18)
```

```
> simplify(f - eval(F, t = 1));  
      0 (19)
```

```
> simplify(PolFromRoots([-1, 1], x) - eval(F, t = 0));  
      0 (20)
```

Une fonction permettant de vérifier qu'un chemin est un bon chemin.

```
> IsGoodPath := proc(F, t, x)  
  local C;  
  C := resultant(F, diff(F, x), x);  
  if (HasRealRoot(C, t, 0, 1) ≠ 1) then  
    RETURN(1);  
  else  
    RETURN(0);  
  end if;  
end;  
> IsGoodPath(F, t, x);  
      1 (21)
```

Une procédure permettant de suivre les racines au fur et à mesure qu'on déforme le polynome.

```
> PathFollowing := proc(Z, f, x, nstep, prec)  
  local F, i, j, δ, Pts;  
  F := Homotop(Z, f, t, x);  
  if (IsGoodPath(F, t, x) = 0) then  
    print("Bad starting point !");  
    RETURN(0);  
  else  
    δ := 1 / nstep;  
    Pts := Z;  
    for i from 1 to nstep do  
      Pts := [seq(MyNewtonMethod(eval(F, t = i·δ), x, Pts[j], prec), j = 1  
        .. nops(Pts))];  
    od;  
    RETURN(Pts);  
  end if;  
end;  
> PathFollowing([-1, 1], f, x, 20, 0.0001);  
      "Bad starting point !" (22)  
      0
```

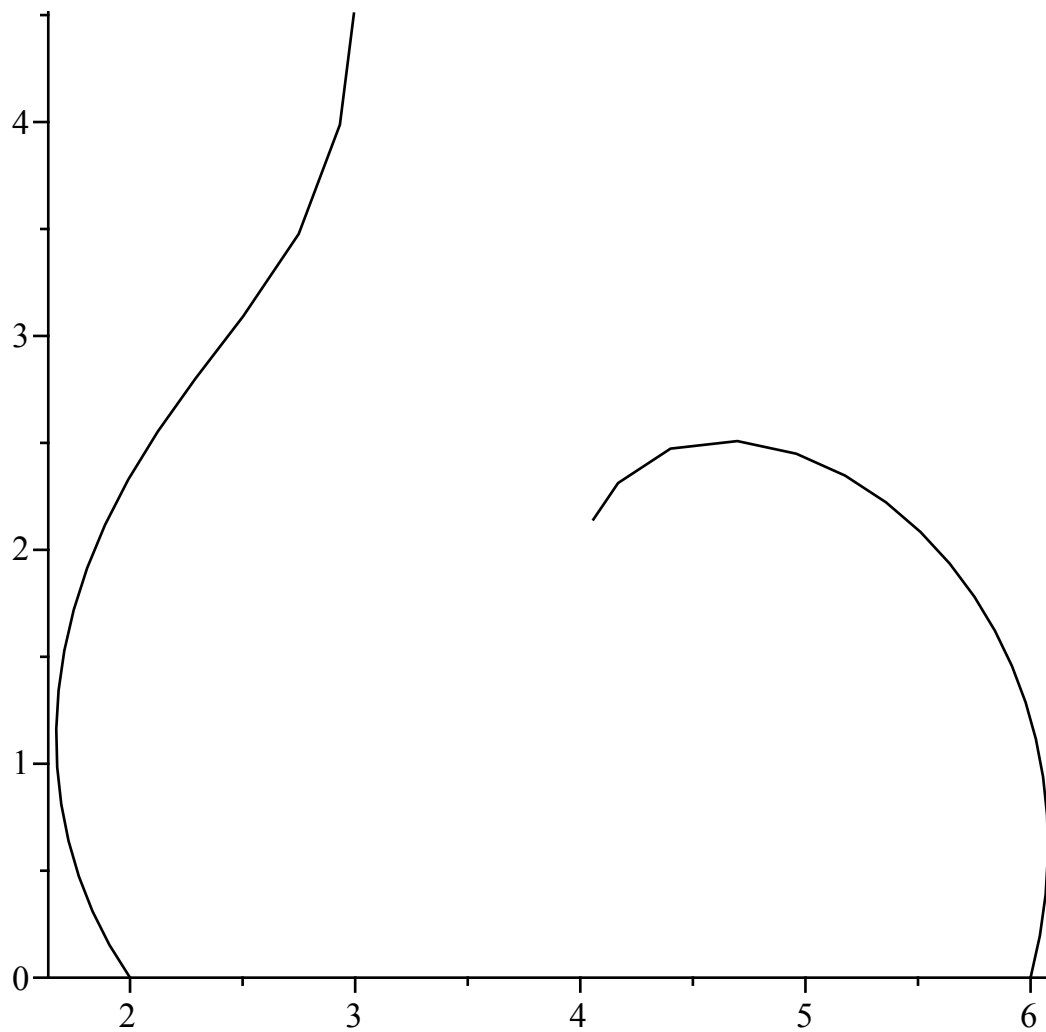
```
> f := PolFromRoots([2, 6], x);  
      f := 12 - 8x + x^2 (23)
```

```
> PathFollowing([-1, 1], f, x, 20, 0.0001);  
      "Bad starting point !"
```

```
> PathFollowing ([3, 4], f, x, 20, 0.0001);
[2.0000000000, 5.9999999999]
```

Le même fonction mais qui retourne une structure de graphe affichable représentant les chemins parcourus par les racines.

```
> GraphPathFollowing := proc (Z, f, x, nstep, prec)
  local F, i, j,  $\delta$ , Pts, s, Paths;
  F := Homotop (Z, f, t, x);
  if (IsGoodPath (F, t, x) = 0) then
    print ("Bad starting point !");
    RETURN (0);
  else
    print ("Path seem to be ok !");
     $\delta := \frac{1}{nstep}$ ;
    Pts := Z,
    s := nops (Pts);
    Paths := [seq ([ ], j=1 ..s)];
    for i from 1 to nstep do
      Pts := [seq (MyNewtonMethod (eval (F, t = i· $\delta$ ), x, Pts[j], prec), j=1
        ..nops (Pts))];
      Paths := [seq ([op (Paths[j]), [ $\Re$  (Pts[j]),  $\Im$  (Pts[j]) ]], j=1 ..s)];
    od;
    Paths := [seq (CURVES (Paths[j]), j=1 ..s)];
    RETURN (Paths);
  end if;
end:
> P := GraphPathFollowing ([3 + 5·I, 4 + 2·I], f, x, 20, 0.0001) :
> display (P);
```



```
> f := PolFromRoots ([2 + I, 1 + 2·I, 6], x);
```

$$f := 30I - 23Ix - 18x + 9x^2 + 3Ix^2 - x^3$$

(26)

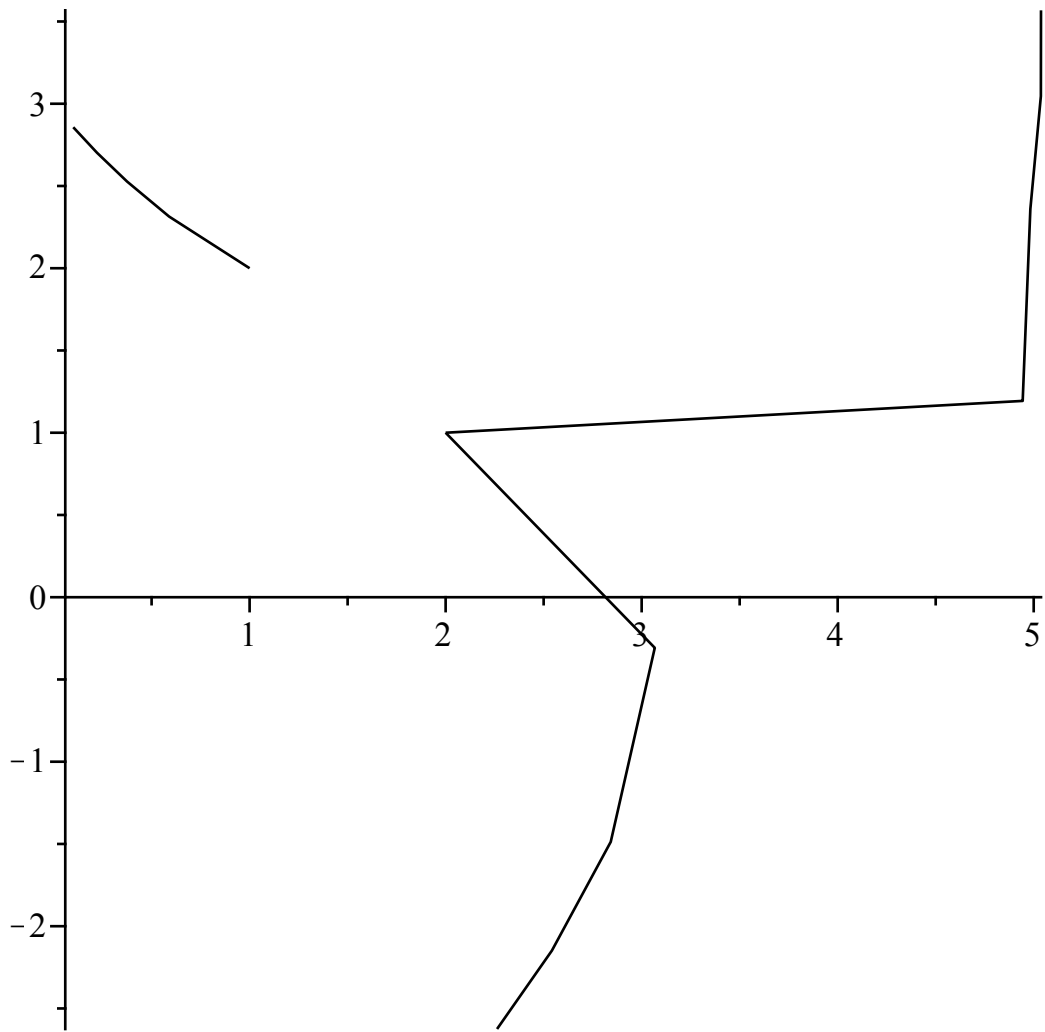
```
> Z := [5 + 4·I, 3·I, 2 - 3·I];
```

$$Z := [5 + 4I, 3I, 2 - 3I]$$

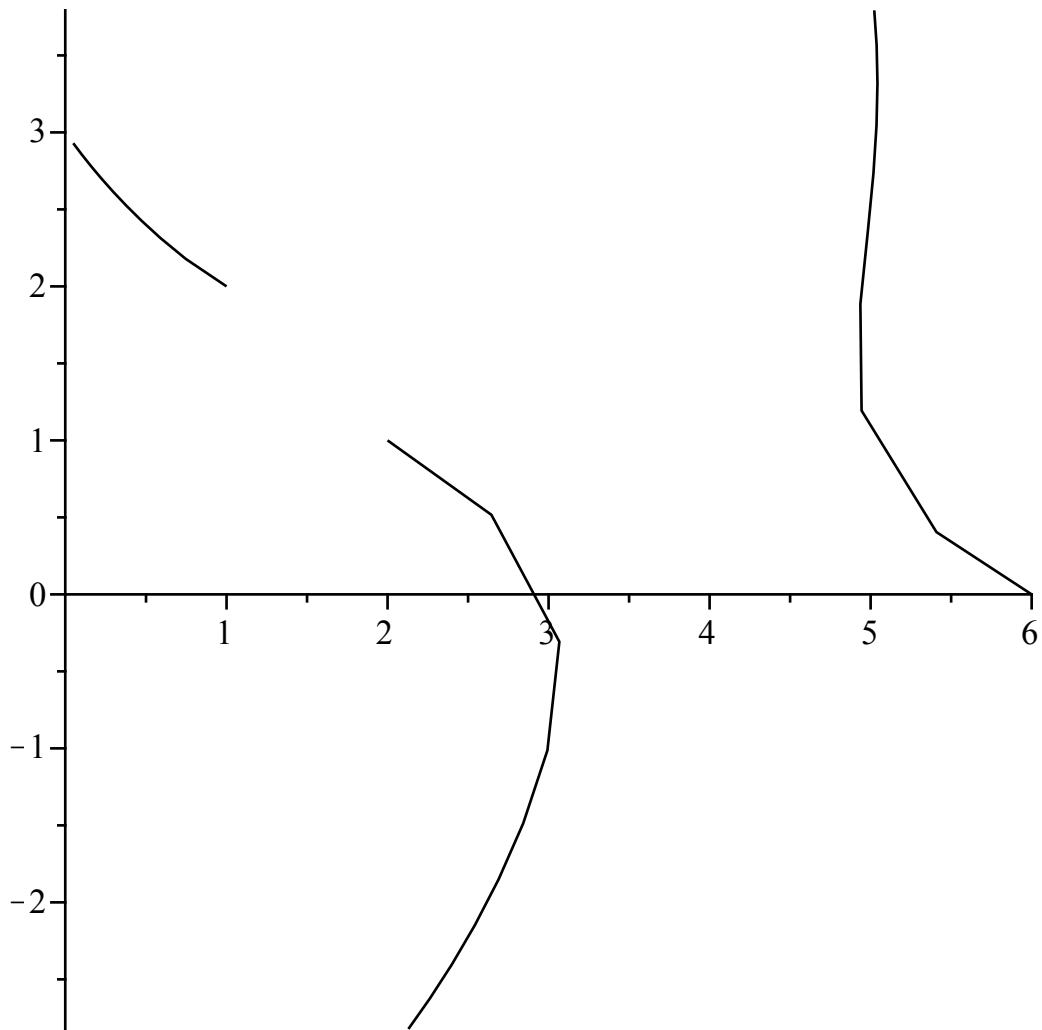
(27)

```
> P := GraphPathFollowing (Z, f, x, 5, 0.0001) :
```

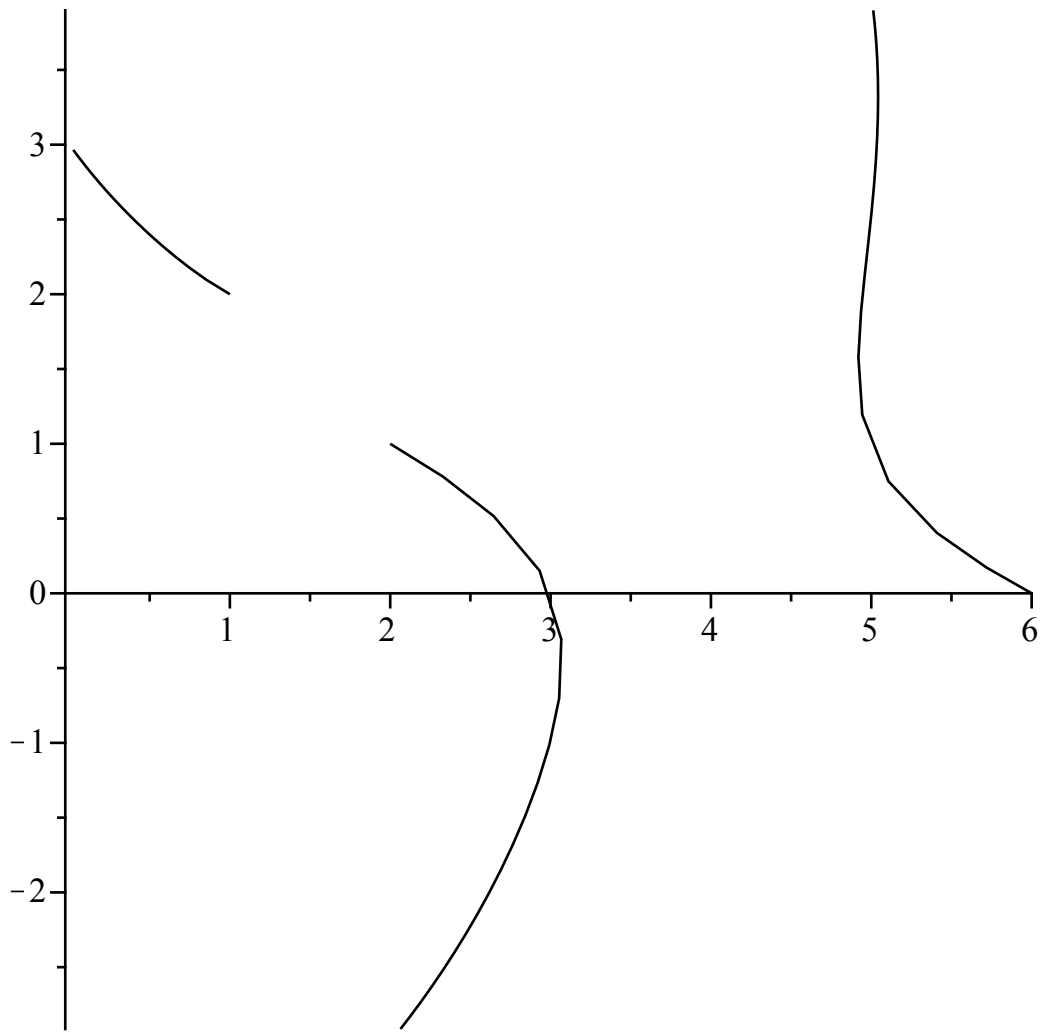
```
> display (P); #ERROR
```



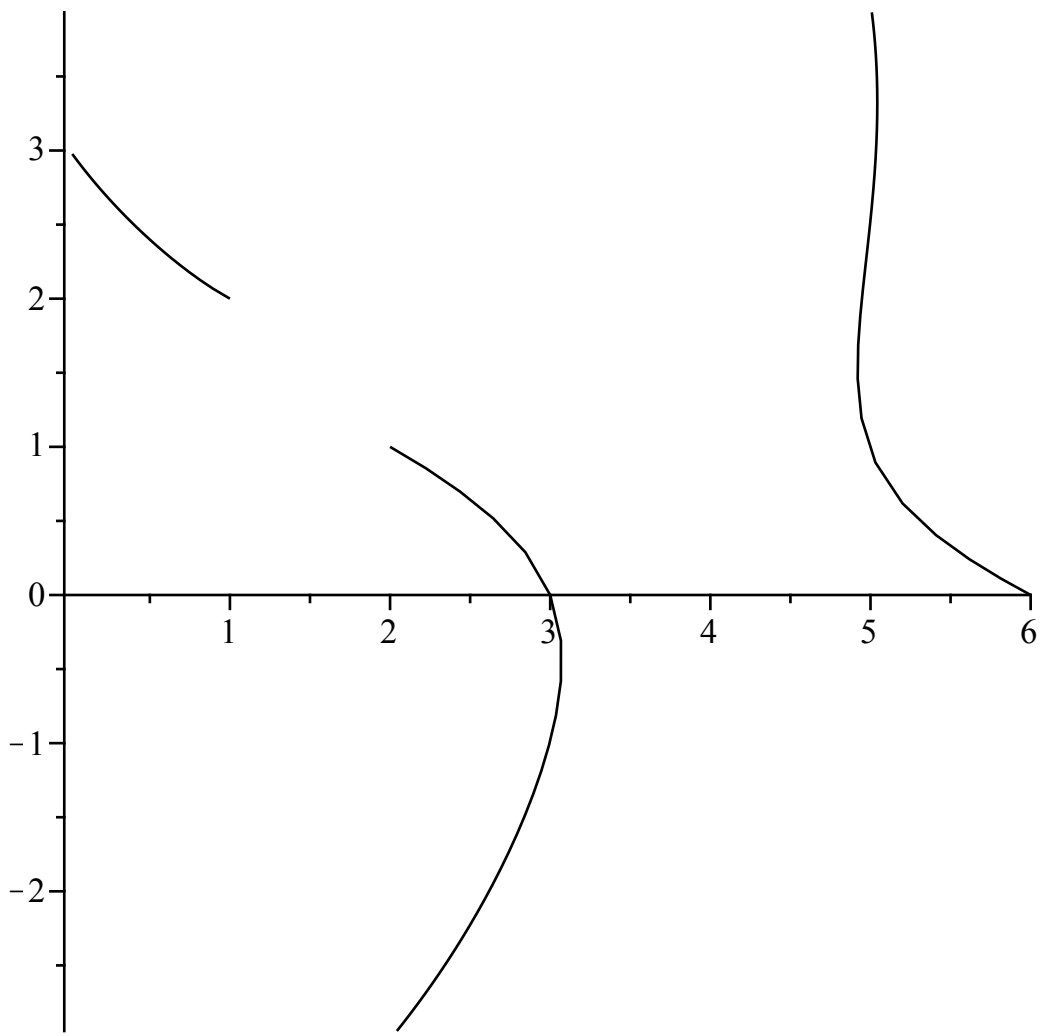
```
> P := GraphPathFollowing(Z, f, x, 10, 0.0001) : display(P); #Now it works
```



```
> P := GraphPathFollowing(Z, f, x, 20, 0.0001) : display(P); #better
```



> $P := \text{GraphPathFollowing}(Z, f, x, 30, 0.0001) : \text{display}(P); \# \text{No win}$



>