5. Die Umkehrfunktion

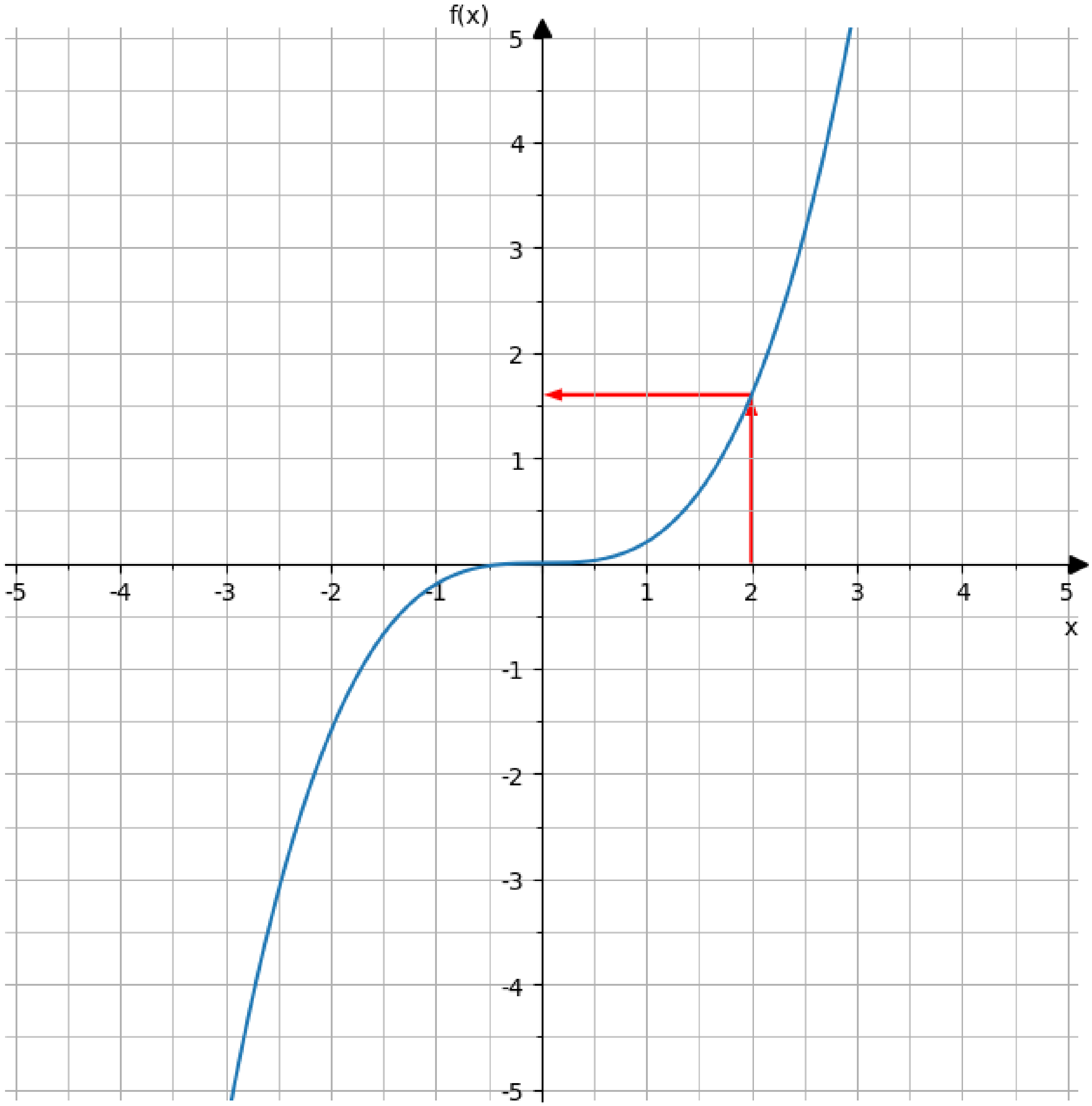
Bisher wurden zu jedem Wert ein f(x)-Wert oder y- Wert zugordnet

Beispiel 1: $f(x) = 0.2 \cdot x^3$

In [33]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import AutoMinorLocator, MultipleLocator, FuncFormatt

# Defintionsmenge und Funktion
# ----------------------------
a= -5.1 # untere x-Intervallgrenze
b= 5.1 # obere x-Intervallgrenze
c = -5.1# untere y-Intervallgrenze
d = 5.1 # obere y-Intervallgrenze
x = np.linspace(a, b,1000)
y1=0.2*x**3
# ----------------------------

# Einstellung des Graphen
fig=plt.figure(figsize=(8,8))
ax = fig.add_subplot(1,1,1, aspect =1)

# Definiton der Haupteinheiten, reele Zahlen ohne die 0
def major_tick(x, pos):
    if x==0:
        return ""
    return int(x)

# Achsenskalierung
ax.xaxis.set_major_locator(MultipleLocator(1))
ax.xaxis.set_minor_locator(AutoMinorLocator(2))
ax.yaxis.set_major_locator(MultipleLocator(1))
ax.yaxis.set_minor_locator(AutoMinorLocator(2))
ax.xaxis.set_major_formatter(FuncFormatter(major_tick))
ax.yaxis.set_major_formatter(FuncFormatter(major_tick))

# Position der Achsen im Schaubild
ax.spines[['top','right']].set_visible(False)
ax.spines[['bottom','left']].set_position('zero')

# Pfeile für die Achsen
ax.plot((1),(0), ls="", marker= ">", ms=7, color="k", transform=ax.get_yaxis
ax.plot((0),(1), ls="", marker= "^", ms=7, color="k", transform=ax.get_xaxis

# Achsenlänge und Beschriftung
ax.set_xlim(a,b)
ax.set_ylim(c, d)
ax.set_xlabel("x", loc="right")
ax.set_ylabel("f(x)", loc="top", rotation=0)
```

```
# Kästchen
ax.grid(linestyle="-", which="major",linewidth=0.7, zorder=-10)
ax.grid(linestyle="-", which="minor",linewidth=0.5, zorder=-10)

# Plot der Funktion
ax.plot(x,y1, zorder=10)
plt.arrow(x=2, y=0, dx=0, dy=1.4,width = 0.04, edgecolor='none', facecolor='
plt.arrow(x=2, y=1.6, dx=-1.8, dy=0,width = 0.04, edgecolor='none', facecolo
#plt.show()
```

Out[33]: <matplotlib.patches.FancyArrow at 0x16a4a6e90>

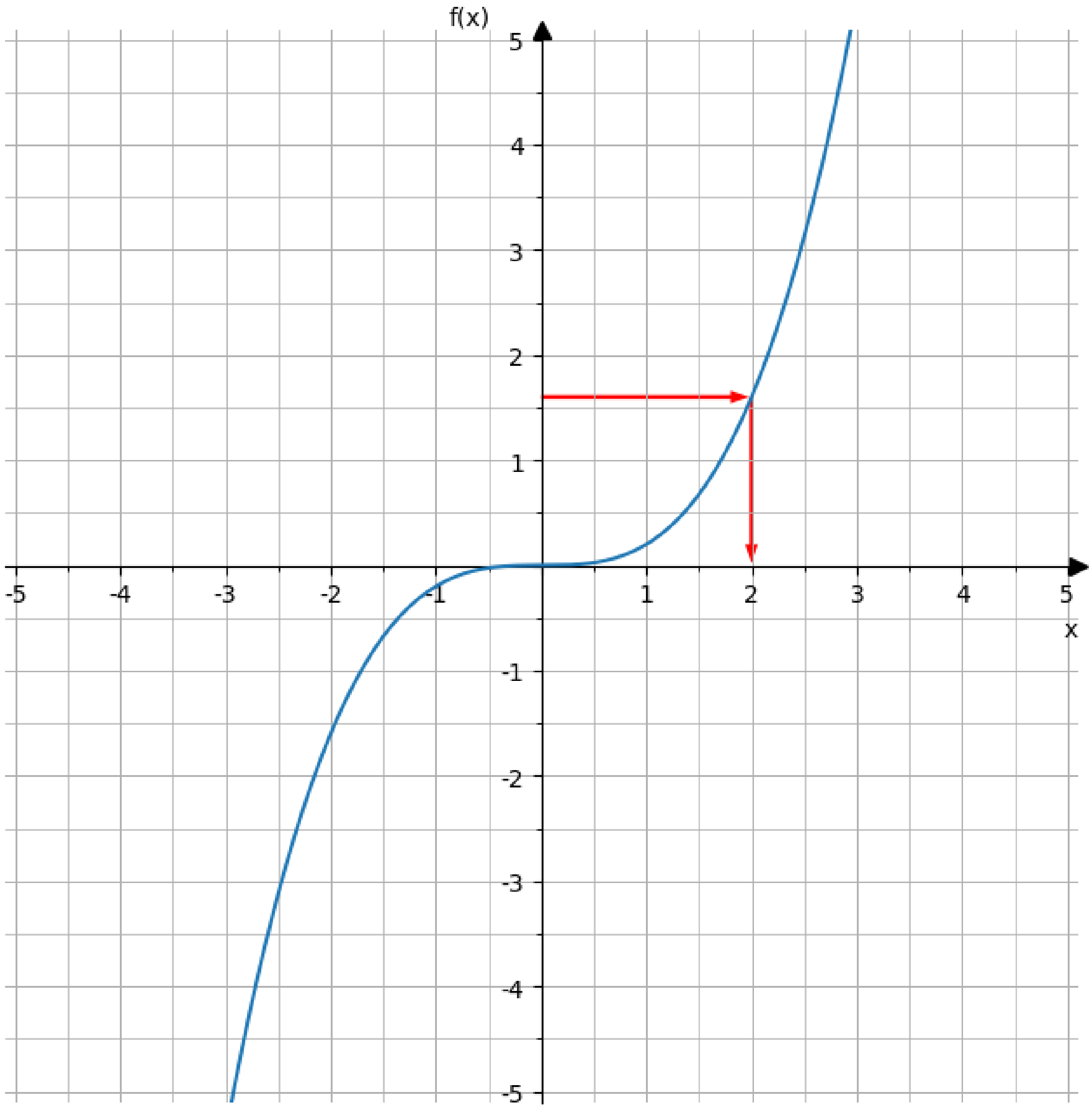


Geht das auch wieder zurück? Kann mann jedem y- Wert wieder den x- Wert zuordnen?

In [34]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import AutoMinorLocator, MultipleLocator, FuncFormatt

# Defintionsmenge und Funktion
```

```python
# -----------------------------
a= -5.1 # untere x-Intervallgrenze
b= 5.1 # obere x-Intervallgrenze
c = -5.1# untere y-Intervallgrenze
d = 5.1 # obere y-Intervallgrenze
x = np.linspace(a, b,1000)
y1=0.2*x**3
# -----------------------------

# Einstellung des Graphen
fig=plt.figure(figsize=(8,8))
ax = fig.add_subplot(1,1,1, aspect =1)

# Definiton der Haupteinheiten, reele Zahlen ohne die 0
def major_tick(x, pos):
    if x==0:
        return ""
    return int(x)

# Achsenskalierung
ax.xaxis.set_major_locator(MultipleLocator(1))
ax.xaxis.set_minor_locator(AutoMinorLocator(2))
ax.yaxis.set_major_locator(MultipleLocator(1))
ax.yaxis.set_minor_locator(AutoMinorLocator(2))
ax.xaxis.set_major_formatter(FuncFormatter(major_tick))
ax.yaxis.set_major_formatter(FuncFormatter(major_tick))

# Position der Achsen im Schaubild
ax.spines[['top','right']].set_visible(False)
ax.spines[['bottom','left']].set_position('zero')

# Pfeile für die Achsen
ax.plot((1),(0), ls="", marker= ">", ms=7, color="k", transform=ax.get_yaxis
ax.plot((0),(1), ls="", marker= "^", ms=7, color="k", transform=ax.get_xaxis

# Achsenlänge und Beschriftung
ax.set_xlim(a,b)
ax.set_ylim(c, d)
ax.set_xlabel("x", loc="right")
ax.set_ylabel("f(x)", loc="top", rotation=0)

# Kästchen
ax.grid(linestyle="-", which="major",linewidth=0.7, zorder=-10)
ax.grid(linestyle="-", which="minor",linewidth=0.5, zorder=-10)

# Plot der Funktion
ax.plot(x,y1, zorder=10)
plt.arrow(x=2, y=1.6, dx=0, dy=-1.4,width = 0.04, edgecolor='none', facecolo
plt.arrow(x=0, y=1.6, dx=1.8, dy=0,width = 0.04, edgecolor='none', facecolor
#plt.show()
```

```
Out[34]: <matplotlib.patches.FancyArrow at 0x172d928d0>
```



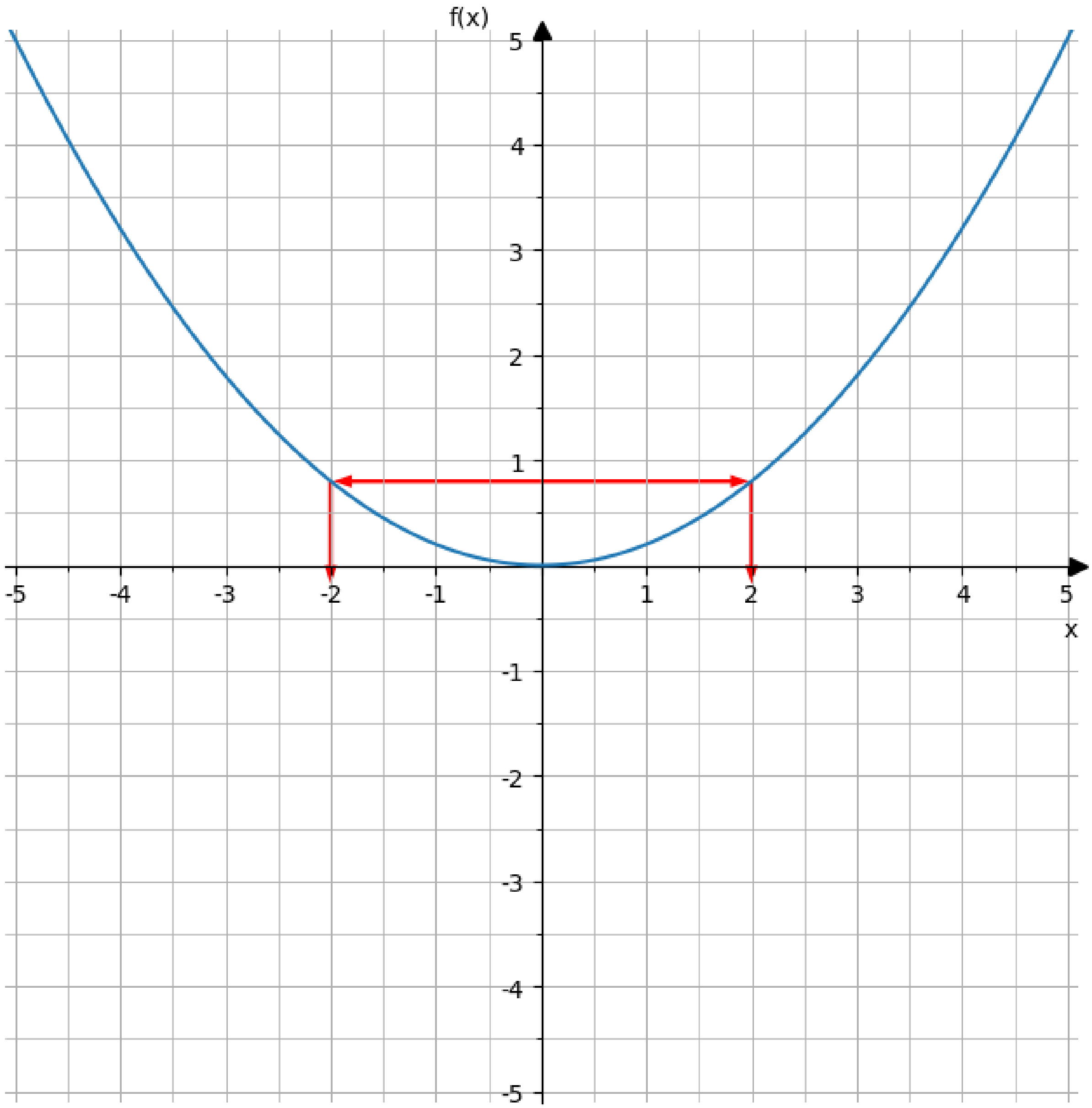
Bei der Funktion $f(x) = 0,2 \cdot x^3$ geht das.

Beispiel 2: $f(x) = 0,2 \cdot x^2$

In [35]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import AutoMinorLocator, MultipleLocator, FuncFormatt

# Defintionsmenge und Funktion
# ----------------------------
a= -5.1 # untere x-Intervallgrenze
b= 5.1 # obere x-Intervallgrenze
c = -5.1# untere y-Intervallgrenze
d = 5.1 # obere y-Intervallgrenze
x = np.linspace(a, b,1000)
y1=0.2*x**2
# ----------------------------
```

```python
# Einstellung des Graphen
fig=plt.figure(figsize=(8,8))
ax = fig.add_subplot(1,1,1, aspect =1)

# Definiton der Haupteinheiten, reele Zahlen ohne die 0
def major_tick(x, pos):
    if x==0:
        return ""
    return int(x)

# Achsenskalierung
ax.xaxis.set_major_locator(MultipleLocator(1))
ax.xaxis.set_minor_locator(AutoMinorLocator(2))
ax.yaxis.set_major_locator(MultipleLocator(1))
ax.yaxis.set_minor_locator(AutoMinorLocator(2))
ax.xaxis.set_major_formatter(FuncFormatter(major_tick))
ax.yaxis.set_major_formatter(FuncFormatter(major_tick))

# Position der Achsen im Schaubild
ax.spines[['top','right']].set_visible(False)
ax.spines[['bottom','left']].set_position('zero')

# Pfeile für die Achsen
ax.plot((1),(0), ls="", marker= ">", ms=7, color="k", transform=ax.get_yaxis
ax.plot((0),(1), ls="", marker= "^", ms=7, color="k", transform=ax.get_xaxis

# Achsenlänge und Beschriftung
ax.set_xlim(a,b)
ax.set_ylim(c, d)
ax.set_xlabel("x", loc="right")
ax.set_ylabel("f(x)", loc="top", rotation=0)

# Kästchen
ax.grid(linestyle="-", which="major",linewidth=0.7, zorder=-10)
ax.grid(linestyle="-", which="minor",linewidth=0.5, zorder=-10)

# Plot der Funktion
ax.plot(x,y1, zorder=10)
plt.arrow(x=2, y=0.8, dx=0, dy=-0.8,width = 0.04, edgecolor='none', facecolo
plt.arrow(x=0, y=0.8, dx=1.8, dy=0,width = 0.04, edgecolor='none', facecolor
plt.arrow(x=-2, y=0.8, dx=0, dy=-0.8,width = 0.04, edgecolor='none', facecol
plt.arrow(x=0, y=0.8, dx=-1.8, dy=0,width = 0.04, edgecolor='none', facecolo
#plt.show()
```

Out[35]:
```
<matplotlib.patches.FancyArrow at 0x16a490b10>
```



Welche Eigenschaften müssen Funktionsgraphen aufweise, damit sie umkehrbar sind?

Beispiele:

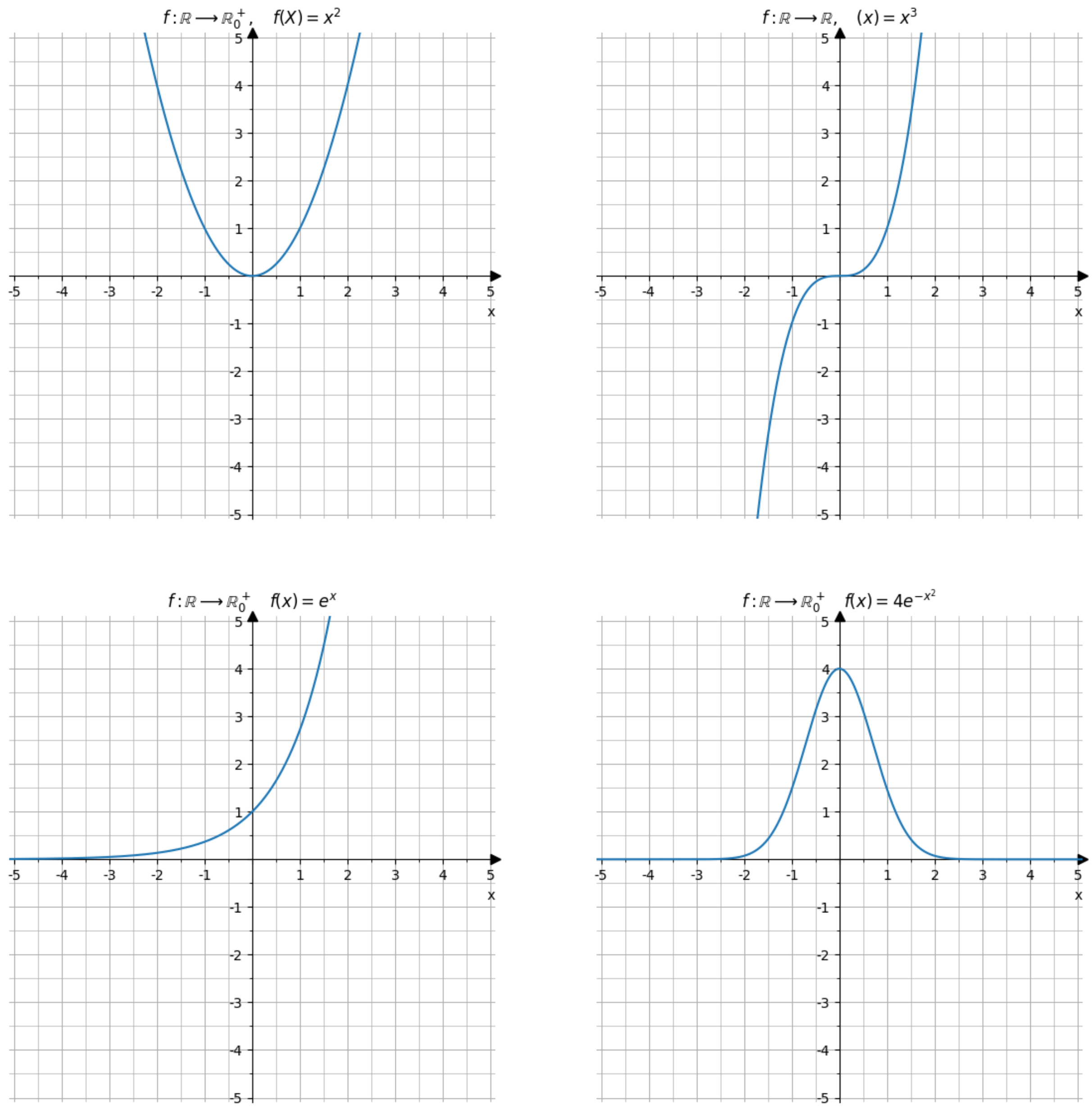
In [36]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import AutoMinorLocator, MultipleLocator, FuncFormatt

# Defintionsmenge und Funktion
# -----------------------------
a= -5.1 # untere x-Intervallgrenze
b= 5.1 # obere x-Intervallgrenze
c = -5.1# untere y-Intervallgrenze
d = 5.1 # obere y-Intervallgrenze
x = np.linspace(a, b,1000)
y1=x**2
y2=x**3
y3=np.exp(x)
```

```python
y4=4*np.exp(-x**2)
# -----------------------------

# Einstellung des Graphen
fig=plt.figure(figsize=(14,14))
ax= fig.add_subplot(2,2,1,aspect=1)
ax1= fig.add_subplot(2,2,2,  aspect=1)
ax2= fig.add_subplot(2,2,3, aspect=1)
ax3= fig.add_subplot(2,2,4, aspect=1)


# Definiton der Haupteinheiten, reele Zahlen ohne die 0
def major_tick(x, pos):
    if x==0:
        return ""
    return int(x)

# Achsenskalierung
ax.xaxis.set_major_locator(MultipleLocator(1))
ax.xaxis.set_minor_locator(AutoMinorLocator(2))
ax.yaxis.set_major_locator(MultipleLocator(1))
ax.yaxis.set_minor_locator(AutoMinorLocator(2))
ax.xaxis.set_major_formatter(FuncFormatter(major_tick))
ax.yaxis.set_major_formatter(FuncFormatter(major_tick))

ax1.xaxis.set_major_locator(MultipleLocator(1))
ax1.xaxis.set_minor_locator(AutoMinorLocator(2))
ax1.yaxis.set_major_locator(MultipleLocator(1))
ax1.yaxis.set_minor_locator(AutoMinorLocator(2))
ax1.xaxis.set_major_formatter(FuncFormatter(major_tick))
ax1.yaxis.set_major_formatter(FuncFormatter(major_tick))

ax2.xaxis.set_major_locator(MultipleLocator(1))
ax2.xaxis.set_minor_locator(AutoMinorLocator(2))
ax2.yaxis.set_major_locator(MultipleLocator(1))
ax2.yaxis.set_minor_locator(AutoMinorLocator(2))
ax2.xaxis.set_major_formatter(FuncFormatter(major_tick))
ax2.yaxis.set_major_formatter(FuncFormatter(major_tick))

ax3.xaxis.set_major_locator(MultipleLocator(1))
ax3.xaxis.set_minor_locator(AutoMinorLocator(2))
ax3.yaxis.set_major_locator(MultipleLocator(1))
ax3.yaxis.set_minor_locator(AutoMinorLocator(2))
ax3.xaxis.set_major_formatter(FuncFormatter(major_tick))
ax3.yaxis.set_major_formatter(FuncFormatter(major_tick))

# Position der Achsen im Schaubild
ax.spines[['top','right']].set_visible(False)
ax.spines[['bottom','left']].set_position('zero')

ax1.spines[['top','right']].set_visible(False)
ax1.spines[['bottom','left']].set_position('zero')

ax2.spines[['top','right']].set_visible(False)
ax2.spines[['bottom','left']].set_position('zero')
```

```
ax3.spines[['top','right']].set_visible(False)
ax3.spines[['bottom','left']].set_position('zero')

# Pfeile für die Achsen
ax.plot((1),(0), ls="", marker= ">", ms=7, color="k", transform=ax.get_yaxis
ax.plot((0),(1), ls="", marker= "^", ms=7, color="k", transform=ax.get_xaxis

ax1.plot((1),(0), ls="", marker= ">", ms=7, color="k", transform=ax1.get_yax
ax1.plot((0),(1), ls="", marker= "^", ms=7, color="k", transform=ax1.get_xax

ax2.plot((1),(0), ls="", marker= ">", ms=7, color="k", transform=ax2.get_yax
ax2.plot((0),(1), ls="", marker= "^", ms=7, color="k", transform=ax2.get_xax

ax3.plot((1),(0), ls="", marker= ">", ms=7, color="k", transform=ax3.get_yax
ax3.plot((0),(1), ls="", marker= "^", ms=7, color="k", transform=ax3.get_xax

# Achsenlänge und Beschriftung
ax.set_xlim(a,b)
ax.set_ylim(c, d)
ax.set_xlabel("x", loc="right")
#ax.set_ylabel("f(x)", loc="top", rotation=0)

ax1.set_xlim(a,b)
ax1.set_ylim(c, d)
ax1.set_xlabel("x", loc="right")
#ax1.set_ylabel("f(x)", loc="top", rotation=0)

ax2.set_xlim(a,b)
ax2.set_ylim(c, d)
ax2.set_xlabel("x", loc="right")
#ax2.set_ylabel("f(x)", loc="top", rotation=0)

ax3.set_xlim(a,b)
ax3.set_ylim(c, d)
ax3.set_xlabel("x", loc="right")
#ax3.set_ylabel("f(x)", loc="top", rotation=0)

# Kästchen
ax.grid(linestyle="-", which="major",linewidth=0.7, zorder=-10)
ax.grid(linestyle="-", which="minor",linewidth=0.5, zorder=-10)

ax1.grid(linestyle="-", which="major",linewidth=0.7, zorder=-10)
ax1.grid(linestyle="-", which="minor",linewidth=0.5, zorder=-10)

ax2.grid(linestyle="-", which="major",linewidth=0.7, zorder=-10)
ax2.grid(linestyle="-", which="minor",linewidth=0.5, zorder=-10)

ax3.grid(linestyle="-", which="major",linewidth=0.7, zorder=-10)
ax3.grid(linestyle="-", which="minor",linewidth=0.5, zorder=-10)

ax.set_title("$f: \mathbb{R} \longrightarrow \mathbb{R}_0^+, \quad f(X)=x^2$
ax1.set_title("$f: \mathbb{R} \longrightarrow \mathbb{R}, \quad (x)=x^3$")
ax2.set_title("$f: \mathbb{R} \longrightarrow \mathbb{R}_0^+ \quad f(x)=e^x$
ax3.set_title("$f: \mathbb{R} \longrightarrow \mathbb{R}_0^+ \quad f(x)=4e^{

# Plot der Funktion
```

```
ax.plot(x,y1, zorder=10)
ax1.plot(x,y2, zorder=10)
ax2.plot(x,y3, zorder=10)
ax3.plot(x,y4, zorder=10)
#plt.show()
```

Out[36]: [<matplotlib.lines.Line2D at 0x17392ee90>]



Definition:
Gegeben ist eine Funktion $f : D \rightarrow W$.

a) Die Funktion f heißt **surjektiv**, wenn es für alle Werte y aus der Wertemenge ($y \in W$) mindestens ein Wert x aus der Definitionsmenge ($x \in D$) gibt, so dass gilt: $f(x) = y$.

b) Die Funkiton f heißt **injektiv**, wenn es für alle Werte x_1, x_2 aus der Definitionsmenge ($x_1, X_2 \in D$) mit der Eigenschaft, dass sie dem gleichen Funktionswert zugeordent werden ($f(x_1) = f(x_2)$) folgt: $x_1 = x_2$.

c) Die Funkiont f heißt **bijektiv**, wenn sie sowohl surjektiv als auch injektiv ist.

Beispiele:

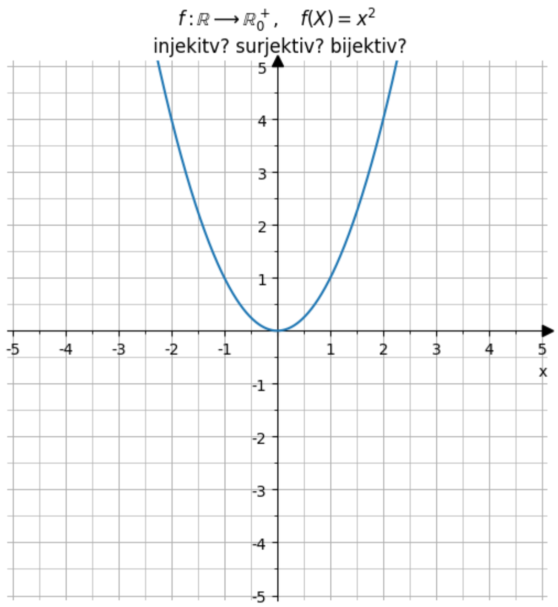
In [37]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import AutoMinorLocator, MultipleLocator, FuncFormatt

# Defintionsmenge und Funktion
# ----------------------------
a= -5.1 # untere x-Intervallgrenze
b= 5.1 # obere x-Intervallgrenze
c = -5.1# untere y-Intervallgrenze
d = 5.1 # obere y-Intervallgrenze
x = np.linspace(a, b,1000)
y1=x**2
y2=x**3
y3=np.exp(x)
y4=4*np.exp(-x**2)
# ----------------------------

# Einstellung des Graphen
fig=plt.figure(figsize=(14,14))
ax= fig.add_subplot(2,2,1,aspect=1)
ax1= fig.add_subplot(2,2,2,  aspect=1)
ax2= fig.add_subplot(2,2,3, aspect=1)
ax3= fig.add_subplot(2,2,4, aspect=1)


# Definiton der Haupteinheiten, reele Zahlen ohne die 0
def major_tick(x, pos):
    if x==0:
        return ""
    return int(x)

# Achsenskalierung
ax.xaxis.set_major_locator(MultipleLocator(1))
ax.xaxis.set_minor_locator(AutoMinorLocator(2))
ax.yaxis.set_major_locator(MultipleLocator(1))
ax.yaxis.set_minor_locator(AutoMinorLocator(2))
ax.xaxis.set_major_formatter(FuncFormatter(major_tick))
ax.yaxis.set_major_formatter(FuncFormatter(major_tick))

ax1.xaxis.set_major_locator(MultipleLocator(1))
ax1.xaxis.set_minor_locator(AutoMinorLocator(2))
ax1.yaxis.set_major_locator(MultipleLocator(1))
ax1.yaxis.set_minor_locator(AutoMinorLocator(2))
ax1.xaxis.set_major_formatter(FuncFormatter(major_tick))
ax1.yaxis.set_major_formatter(FuncFormatter(major_tick))

ax2.xaxis.set_major_locator(MultipleLocator(1))
ax2.xaxis.set_minor_locator(AutoMinorLocator(2))
ax2.yaxis.set_major_locator(MultipleLocator(1))
```

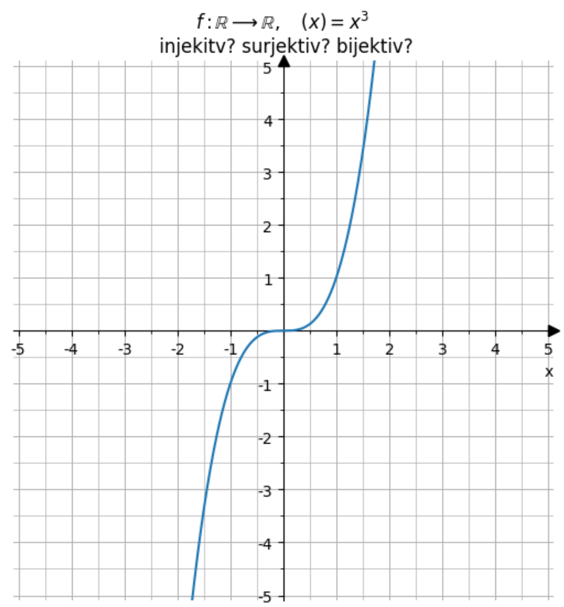
```
ax2.yaxis.set_minor_locator(AutoMinorLocator(2))
ax2.xaxis.set_major_formatter(FuncFormatter(major_tick))
ax2.yaxis.set_major_formatter(FuncFormatter(major_tick))

ax3.xaxis.set_major_locator(MultipleLocator(1))
ax3.xaxis.set_minor_locator(AutoMinorLocator(2))
ax3.yaxis.set_major_locator(MultipleLocator(1))
ax3.yaxis.set_minor_locator(AutoMinorLocator(2))
ax3.xaxis.set_major_formatter(FuncFormatter(major_tick))
ax3.yaxis.set_major_formatter(FuncFormatter(major_tick))

# Position der Achsen im Schaubild
ax.spines[['top','right']].set_visible(False)
ax.spines[['bottom','left']].set_position('zero')

ax1.spines[['top','right']].set_visible(False)
ax1.spines[['bottom','left']].set_position('zero')

ax2.spines[['top','right']].set_visible(False)
ax2.spines[['bottom','left']].set_position('zero')

ax3.spines[['top','right']].set_visible(False)
ax3.spines[['bottom','left']].set_position('zero')

# Pfeile für die Achsen
ax.plot((1),(0), ls="", marker= ">", ms=7, color="k", transform=ax.get_yaxis
ax.plot((0),(1), ls="", marker= "^", ms=7, color="k", transform=ax.get_xaxis

ax1.plot((1),(0), ls="", marker= ">", ms=7, color="k", transform=ax1.get_yax
ax1.plot((0),(1), ls="", marker= "^", ms=7, color="k", transform=ax1.get_xax

ax2.plot((1),(0), ls="", marker= ">", ms=7, color="k", transform=ax2.get_yax
ax2.plot((0),(1), ls="", marker= "^", ms=7, color="k", transform=ax2.get_xax

ax3.plot((1),(0), ls="", marker= ">", ms=7, color="k", transform=ax3.get_yax
ax3.plot((0),(1), ls="", marker= "^", ms=7, color="k", transform=ax3.get_xax

# Achsenlänge und Beschriftung
ax.set_xlim(a,b)
ax.set_ylim(c, d)
ax.set_xlabel("x", loc="right")
#ax.set_ylabel("f(x)", loc="top", rotation=0)

ax1.set_xlim(a,b)
ax1.set_ylim(c, d)
ax1.set_xlabel("x", loc="right")
#ax1.set_ylabel("f(x)", loc="top", rotation=0)

ax2.set_xlim(a,b)
ax2.set_ylim(c, d)
ax2.set_xlabel("x", loc="right")
#ax2.set_ylabel("f(x)", loc="top", rotation=0)

ax3.set_xlim(a,b)
ax3.set_ylim(c, d)
ax3.set_xlabel("x", loc="right")
```

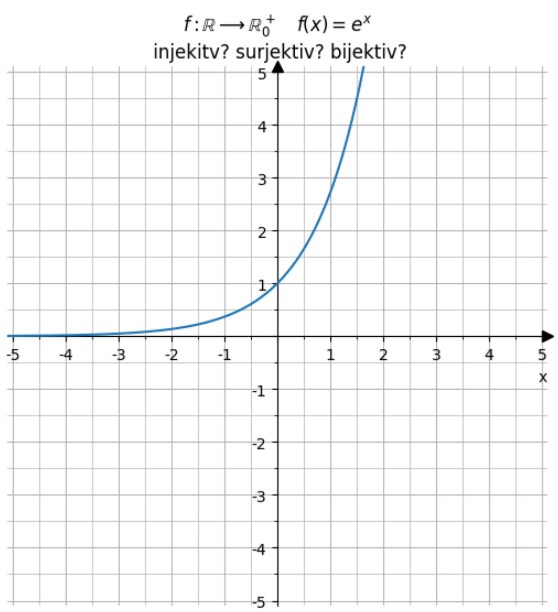
```
#ax3.set_ylabel("f(x)", loc="top", rotation=0)

# Kästchen
ax.grid(linestyle="-", which="major",linewidth=0.7, zorder=-10)
ax.grid(linestyle="-", which="minor",linewidth=0.5, zorder=-10)

ax1.grid(linestyle="-", which="major",linewidth=0.7, zorder=-10)
ax1.grid(linestyle="-", which="minor",linewidth=0.5, zorder=-10)

ax2.grid(linestyle="-", which="major",linewidth=0.7, zorder=-10)
ax2.grid(linestyle="-", which="minor",linewidth=0.5, zorder=-10)

ax3.grid(linestyle="-", which="major",linewidth=0.7, zorder=-10)
ax3.grid(linestyle="-", which="minor",linewidth=0.5, zorder=-10)

ax.set_title("$f: \mathbb{R} \longrightarrow \mathbb{R}_0^+, \quad f(X)=x^2$
ax1.set_title("$f: \mathbb{R} \longrightarrow \mathbb{R}, \quad (x)=x^3$ \n
ax2.set_title("$f: \mathbb{R} \longrightarrow \mathbb{R}_0^+ \quad f(x)=e^x$
ax3.set_title("$f: \mathbb{R} \longrightarrow \mathbb{R}_0^+ \quad f(x)=4e^{

# Plot der Funktion

ax.plot(x,y1, zorder=10)
ax1.plot(x,y2, zorder=10)
ax2.plot(x,y3, zorder=10)
ax3.plot(x,y4, zorder=10)
#plt.show()
```

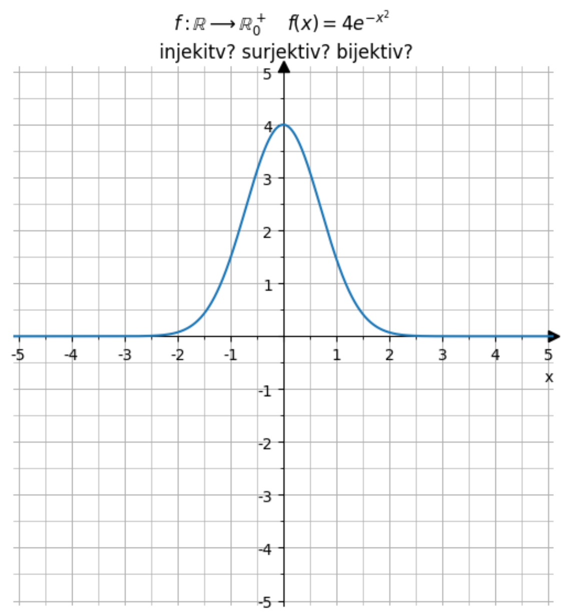
Out[37]:

```
[<matplotlib.lines.Line2D at 0x173c78610>]
```









Satz:

Gegeben ist eine differenzierbare Funktion $f: D \to W$.

1. f injektiv $\Leftrightarrow$ $f'(x) > 0$ für alle $x \in D$ (f streng mono st.)
2. f injektiv $\Leftrightarrow$ $f'(x) < 0$ für alle $x \in D$(f streng mono fa.)

Bemerkung:
Die strenge Monotonie ist auch erfüllt, wenn für ein $x \in D$, welches auf dem Rand von D liegt, gilt:

- $f'(x) = 0$ oder
- $f'(x)$ ist nicht definiert.

Beweis:
Sei $f : D \to W$ gegeben mit:

- f ist differenzierbar $\Rightarrow$ f stetig
- f injektiv

1. Zeige: "$\Rightarrow$"
   f ist genau dann streng monoton wachsend, wenn für drei verschieden Punkte $(a|f(a)), (x|f(x)), (b|f(b))$ mit $a, x, b \in D$ mit $a < x < b$ gilt:
   $f(a) < f(x) < f(b)$
   Die offenen Intervalle $(f(a); f(x)))$ und $(f(x); f(b))$ überschneiden sich nicht (sind disjunkt)

   Angenommen es gibt ein $x_0 \in (a, b) \subset D$, dessen Funktionswert $f(x_0)$ nicht zwischen $f(a)$ und $f(b)$ liegt.

   aus dem Zwischenwertsatz folgt dann:

   - Jeder Wert zwischen $f(x_0)$ und $f(a)$ wird einmal auf dem Intervall (a, x_0) angenommen.
   - Jeder Wert zwischen $f(x_0)$ und $f(b)$ wird einmal auf dem Intervall (x_0, b) angenommen. Widerspruch zur Annahme f sei injektiv.

   f ist streng monoton fallend, ... Analog.

2. Zeige: "$\Leftarrow$"
   Jede streng monoton wachsende bzw fallende Funktion ist injektiv. $\square$

Zwischenwertsatz:

Sei $f : [a, b] \to (R)$ eine stetige Funktion mit $f(a) < 0$ und $f(b) > 0$, (bzw. $f(a) > 0$ und $f(b) < 0$). Dann exisitert ein $p \in [a, b]$ mit $f(p) = 0$.

Bemerkung:

- Der Zwischenwertsatz lässt sich auf beliebiges f(p)=c übertragen.
- Anschaulich sagt der Zwischenwertsatz dann folgendes aus: Wenn man eine stetige Funktion auf einem Intervall $[a, b]$ hat, die die Geraden $y = f(a)$ und $y = f(b)$ schneidet, dann muss diese Funktion zwangsläufig auch alle anderen Geraden, die parallel zur x-Achse liegen, schneiden, die sich zwischen f(a) und f(b) befinden.

Corollar:

Sei $f : [a, b] \to (R)$ eine stetige Funktion und c eine reelle Zahl zwischen $f(a)$ und $f(b)$. Dann exisitert ein $p \in [a, b]$ mit $f(p) = c$.

Definition:

Eine Funktion $f: D \to W, f(x) = y$ heißt umkehrbar, wenn f injektiv ist.
Die Umkehrfunktion wird definiert durch: $f^{-1}(y) = x$, für $y \in W$.
f^{-1} hat als Defintionsmenge W und als Wertemenge D.

Es gilt:
$f^{-1}(f(x)) = x$ für alle $x \in D$
$f(f^{-1}(x)) = x$ für alle $x \in D$

Algorithmus zur Bestimmung der Umkehrfunktion:

1. Prüfe, ob f umkehrbar ist, d.h. ob f injekitv oder bijektiv ist.
2. Setze $y = f(x)$
3. Löse Gleichung nach x auf.
4. Vertausche x und y.
5. Erstze y durch f^{-1}

Beispiel:

Gegeben ist $f: [-1, \infty) \to [0; \infty), f(x) = (x+1)^2$

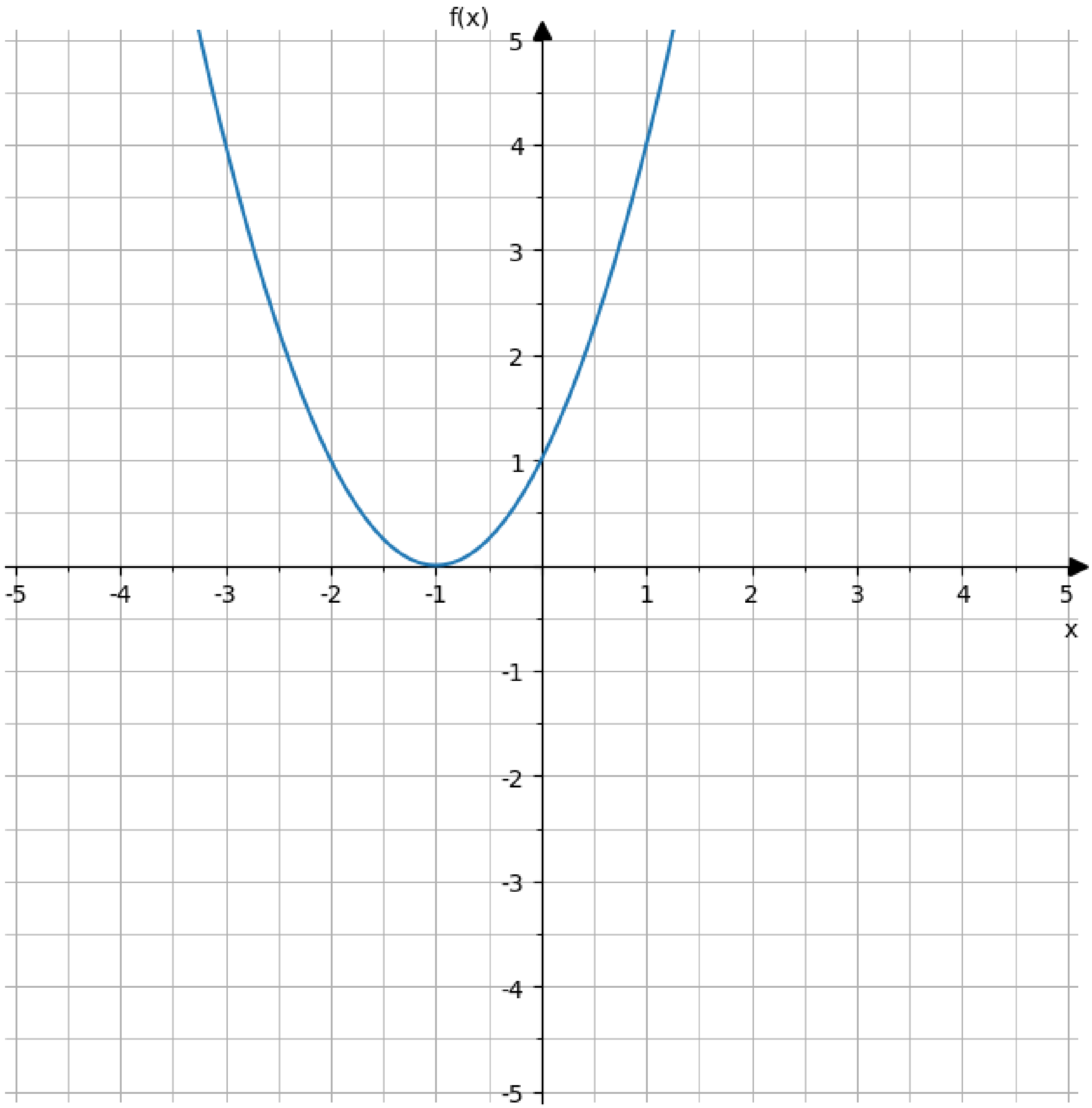
1. Prüfung auf Injektivität.

In [38]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import AutoMinorLocator, MultipleLocator, FuncFormatt

# Defintionsmenge und Funktion
# ----------------------------
a= -5.1 # untere x-Intervallgrenze
b= 5.1 # obere x-Intervallgrenze
c = -5.1# untere y-Intervallgrenze
d = 5.1 # obere y-Intervallgrenze
x = np.linspace(a, b,1000)
y1=(x+1)**2
# ----------------------------

# Einstellung des Graphen
fig=plt.figure(figsize=(8,8))
ax= fig.add_subplot(1,1,1, aspect =1)

# Definiton der Haupteinheiten, reele Zahlen ohne die 0
def major_tick(x, pos):
    if x==0:
        return ""
    return int(x)

# Achsenskalierung
ax.xaxis.set_major_locator(MultipleLocator(1))
ax.xaxis.set_minor_locator(AutoMinorLocator(2))
ax.yaxis.set_major_locator(MultipleLocator(1))
ax.yaxis.set_minor_locator(AutoMinorLocator(2))
```

```
ax.xaxis.set_major_formatter(FuncFormatter(major_tick))
ax.yaxis.set_major_formatter(FuncFormatter(major_tick))

# Position der Achsen im Schaubild
ax.spines[['top','right']].set_visible(False)
ax.spines[['bottom','left']].set_position('zero')

# Pfeile für die Achsen
ax.plot((1),(0), ls="", marker= ">", ms=7, color="k", transform=ax.get_yaxis
ax.plot((0),(1), ls="", marker= "^", ms=7, color="k", transform=ax.get_xaxis

# Achsenlänge und Beschriftung
ax.set_xlim(a,b)
ax.set_ylim(c, d)
ax.set_xlabel("x", loc="right")
ax.set_ylabel("f(x)", loc="top", rotation=0)

# Kästchen
ax.grid(linestyle="-", which="major",linewidth=0.7, zorder=-10)
ax.grid(linestyle="-", which="minor",linewidth=0.5, zorder=-10)

# Plot der Funktion
ax.plot(x,y1, zorder=10)
#plt.show()
```

Out[38]:

```
[<matplotlib.lines.Line2D at 0x1683e97d0>]
```



1. $f'(x) = 2(x+1), \quad 2(x+1) > 0$ wenn $x > -1 \Rightarrow f$ ist auf jedenfall injektiv und damit umkehrbar.
2. $y = (x+1)^2$
3. $x_1 = \sqrt{y-1}$ und $x_2 = -\sqrt{y} - 1$
   Da $x \in [-1; \infty)$ betrachten wir nur x_1
4. $y = \sqrt{x} - 1$
5. $f^{-1}(x) = \sqrt{x} - 1$

Der Graph von Umkehrfunktionen:

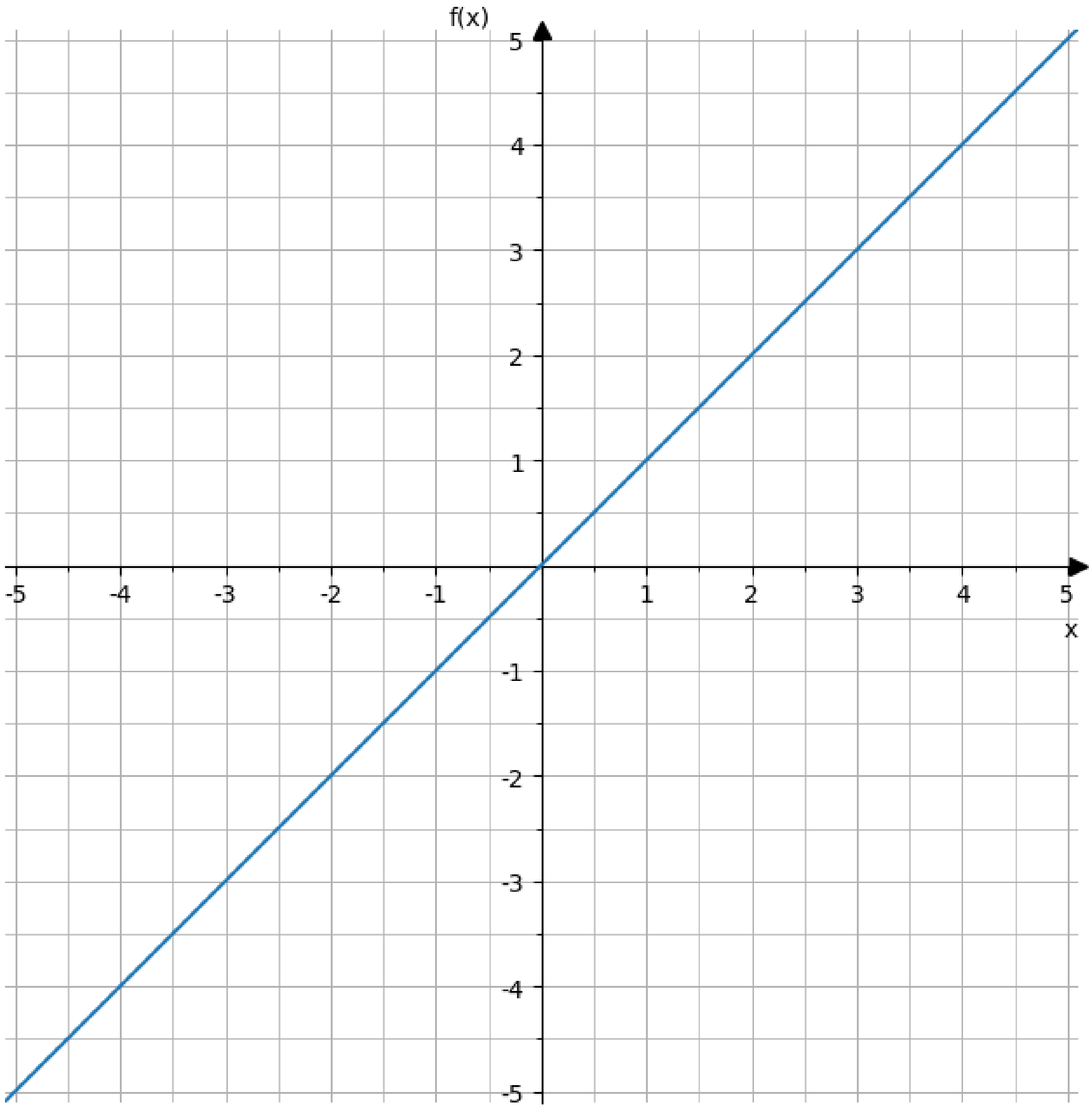
Beispiel 1: $f(x) = x$

In [39]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import AutoMinorLocator, MultipleLocator, FuncFormatt
```

```
# Defintionsmenge und Funktion
# ----------------------------
a= -5.1 # untere x-Intervallgrenze
b= 5.1 # obere x-Intervallgrenze
c = -5.1# untere y-Intervallgrenze
d = 5.1 # obere y-Intervallgrenze
x = np.linspace(a, b,1000)
y1=x
# ----------------------------

# Einstellung des Graphen
fig=plt.figure(figsize=(8,8))
ax= fig.add_subplot(1,1,1, aspect =1)

# Definiton der Haupteinheiten, reele Zahlen ohne die 0
def major_tick(x, pos):
    if x==0:
        return ""
    return int(x)

# Achsenskalierung
ax.xaxis.set_major_locator(MultipleLocator(1))
ax.xaxis.set_minor_locator(AutoMinorLocator(2))
ax.yaxis.set_major_locator(MultipleLocator(1))
ax.yaxis.set_minor_locator(AutoMinorLocator(2))
ax.xaxis.set_major_formatter(FuncFormatter(major_tick))
ax.yaxis.set_major_formatter(FuncFormatter(major_tick))

# Position der Achsen im Schaubild
ax.spines[['top','right']].set_visible(False)
ax.spines[['bottom','left']].set_position('zero')

# Pfeile für die Achsen
ax.plot((1),(0), ls="", marker= ">", ms=7, color="k", transform=ax.get_yaxis
ax.plot((0),(1), ls="", marker= "^", ms=7, color="k", transform=ax.get_xaxis

# Achsenlänge und Beschriftung
ax.set_xlim(a,b)
ax.set_ylim(c, d)
ax.set_xlabel("x", loc="right")
ax.set_ylabel("f(x)", loc="top", rotation=0)

# Kästchen
ax.grid(linestyle="-", which="major",linewidth=0.7, zorder=-10)
ax.grid(linestyle="-", which="minor",linewidth=0.5, zorder=-10)

# Plot der Funktion
ax.plot(x,y1, zorder=10)
#plt.show()
```

Out[39]:

```
[<matplotlib.lines.Line2D at 0x173d691d0>]
```

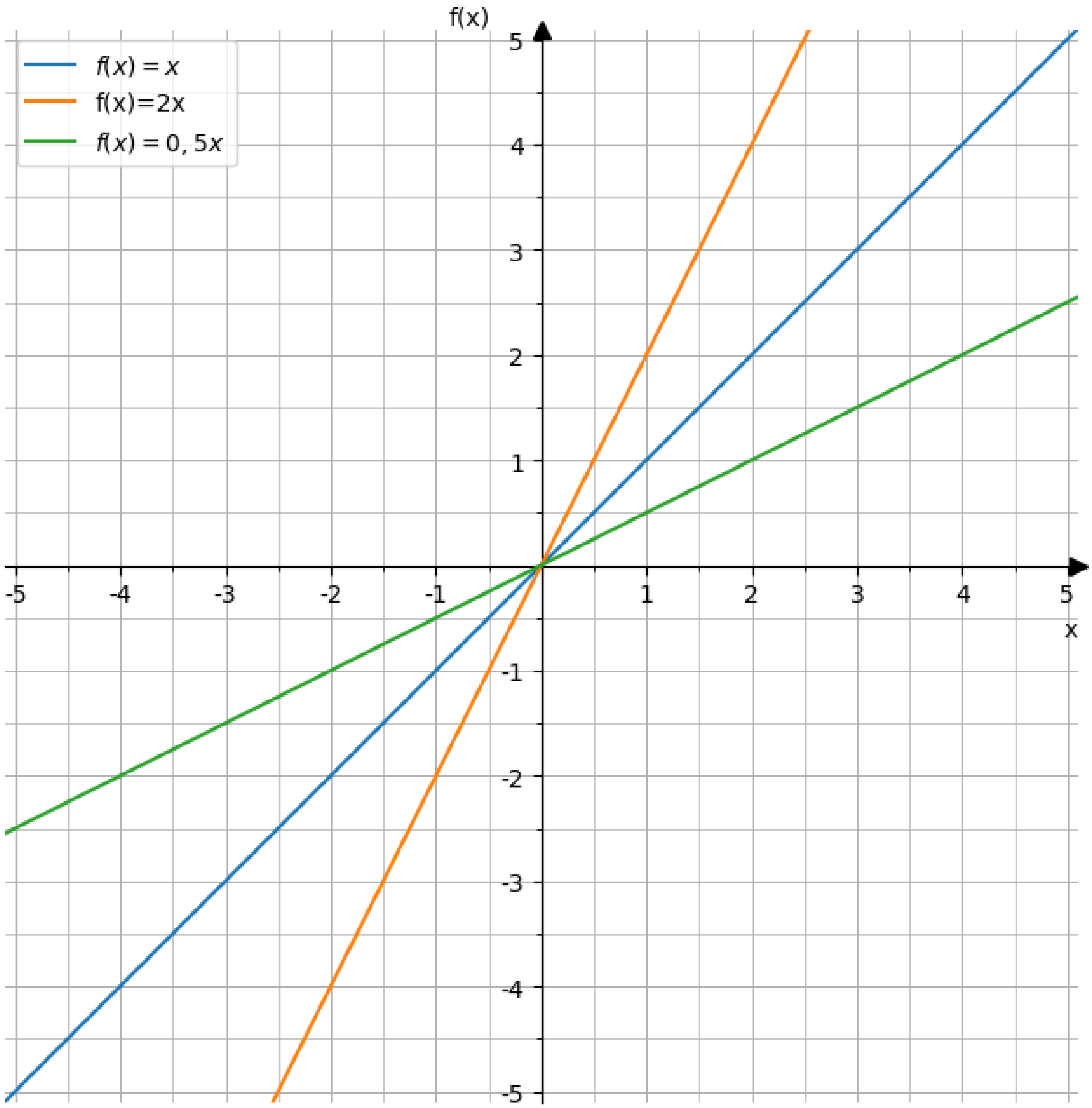


Beispiel 1: $f(x) = 2x$

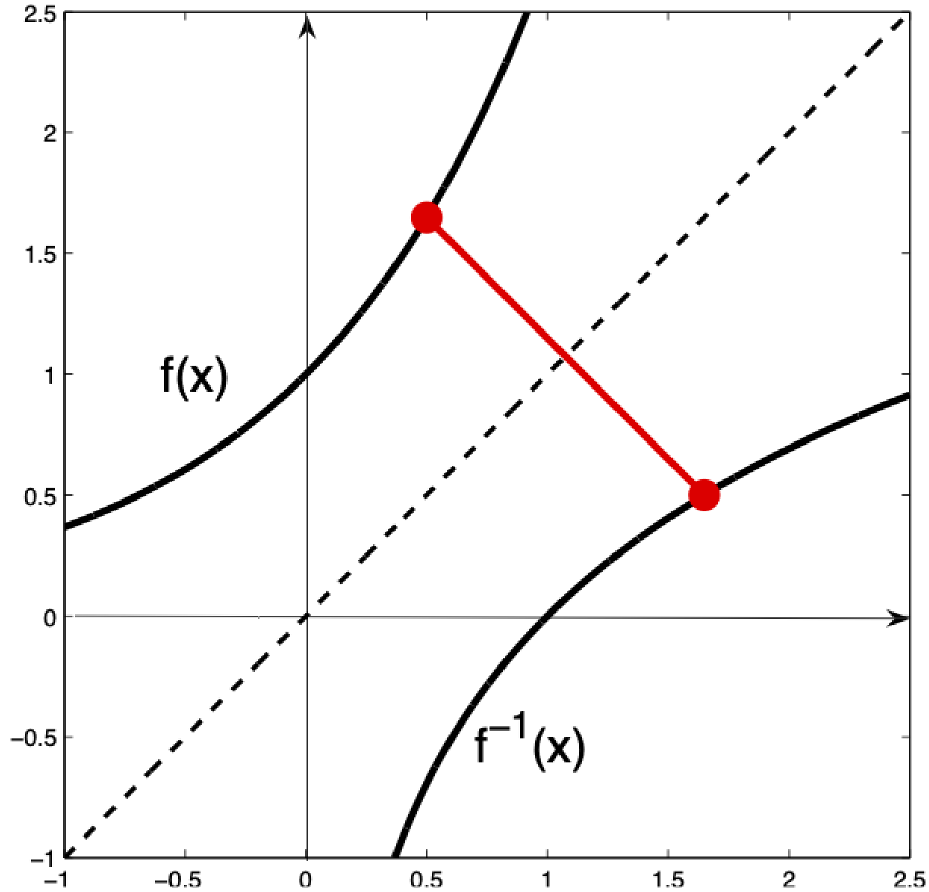
In [40]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import AutoMinorLocator, MultipleLocator, FuncFormatt

# Defintionsmenge und Funktion
# ----------------------------
a= -5.1 # untere x-Intervallgrenze
b= 5.1 # obere x-Intervallgrenze
c = -5.1# untere y-Intervallgrenze
d = 5.1 # obere y-Intervallgrenze
x = np.linspace(a, b,1000)
y1=x
y2=2*x
y3=0.5*x
# ----------------------------
```

```python
# Einstellung des Graphen
fig=plt.figure(figsize=(8,8))
ax= fig.add_subplot(1,1,1, aspect =1)

# Definiton der Haupteinheiten, reele Zahlen ohne die 0
def major_tick(x, pos):
    if x==0:
        return ""
    return int(x)

# Achsenskalierung
ax.xaxis.set_major_locator(MultipleLocator(1))
ax.xaxis.set_minor_locator(AutoMinorLocator(2))
ax.yaxis.set_major_locator(MultipleLocator(1))
ax.yaxis.set_minor_locator(AutoMinorLocator(2))
ax.xaxis.set_major_formatter(FuncFormatter(major_tick))
ax.yaxis.set_major_formatter(FuncFormatter(major_tick))

# Position der Achsen im Schaubild
ax.spines[['top','right']].set_visible(False)
ax.spines[['bottom','left']].set_position('zero')

# Pfeile für die Achsen
ax.plot((1),(0), ls="", marker= ">", ms=7, color="k", transform=ax.get_yaxis
ax.plot((0),(1), ls="", marker= "^", ms=7, color="k", transform=ax.get_xaxis

# Achsenlänge und Beschriftung
ax.set_xlim(a,b)
ax.set_ylim(c, d)
ax.set_xlabel("x", loc="right")
ax.set_ylabel("f(x)", loc="top", rotation=0)


# Kästchen
ax.grid(linestyle="-", which="major",linewidth=0.7, zorder=-10)
ax.grid(linestyle="-", which="minor",linewidth=0.5, zorder=-10)


# Plot der Funktion
ax.plot(x,y1, zorder=10, label = "$f(x)=x$")
ax.plot(x,y2, zorder=10, label= "f(x)=2x")
ax.plot(x,y3, zorder=10, label = "$f(x)=0,5x$")
ax.legend()
#plt.show()
```

Out[40]: <matplotlib.legend.Legend at 0x16a201690>



Bemerkung: Der Graph von f^{-1} entsteht aus dem Graphen von f durch Spiegelung an der 1. Winkelhalbierenden ($y = x$).



f(x)
$f^{-1}(x)$