» Gradient Descent

* Recall general iterative minimisation algorithm:

```
x=x0
for k in range(num_iters):
    step = calcStep(fn,x)
    x = x - step
```

* We know one way to choose the step, namely:

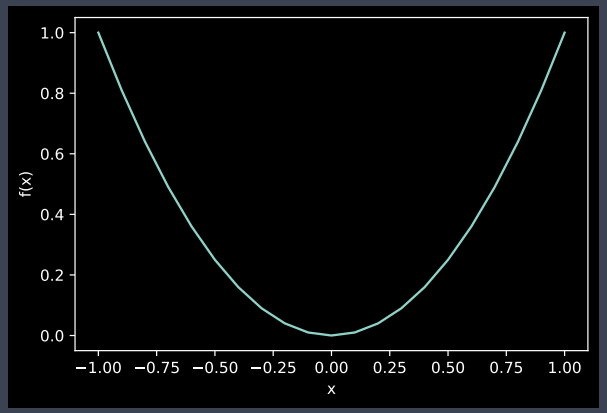
$$step = \alpha[\frac{\partial f}{\partial x_1}(x), \frac{\partial f}{\partial x_2}(x), \ldots, \frac{\partial f}{\partial x_n}(x)]$$

where α is the *step size* or *learning rate*

* How to choose α?

» Example

* Minimise function $f(x) = x^2$. Derivative $df/dx = 2x$.

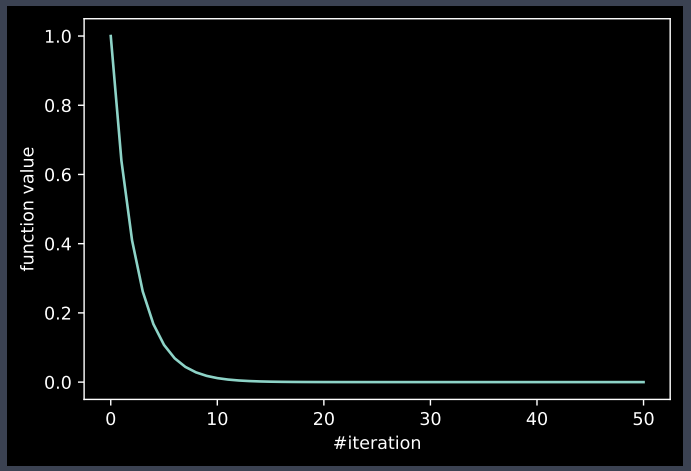


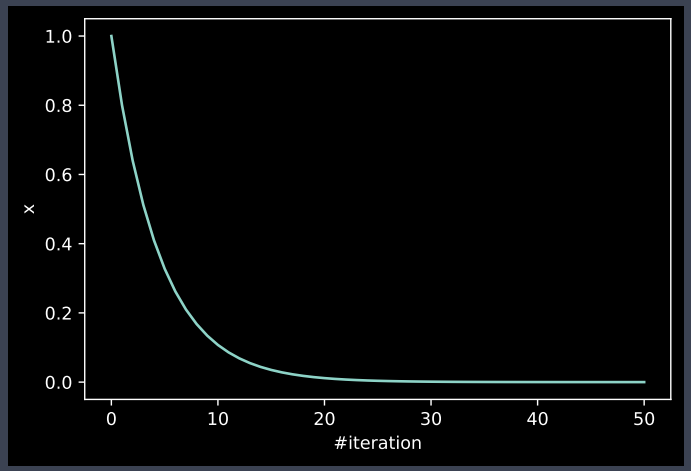
* Minimum of $f(x) = 0$ when $x = 0$
* Python code:

```
class QuadraticFn():
    def f(self, x):
        return x**2 # function value f(x)

    def df(self, x):
        return 2*x # derivative of f(x)

def gradDescent(fn,x0,alpha=0.15,num_iters=50):
    x=x0;
    X=np.array([x]); F=np.array(fn.f(x));
    for k in range(num_iters):
        step = alpha*fn.df(x)
        x = x - step
        X=np.append(X,[x],axis=0); F=np.append(F,fn.f(x))
    return (X,F)

(X, F) = gradDescent(fn,x0=1,alpha=0.1)
```

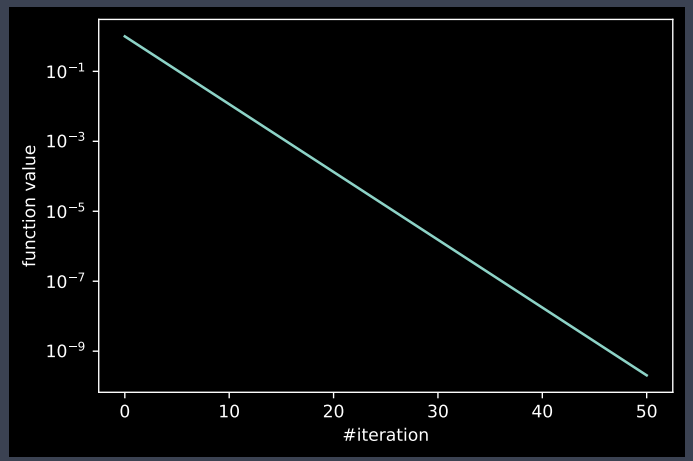
» Example

* Starting value $x0 = 1$, step size $alpha = 0.1$





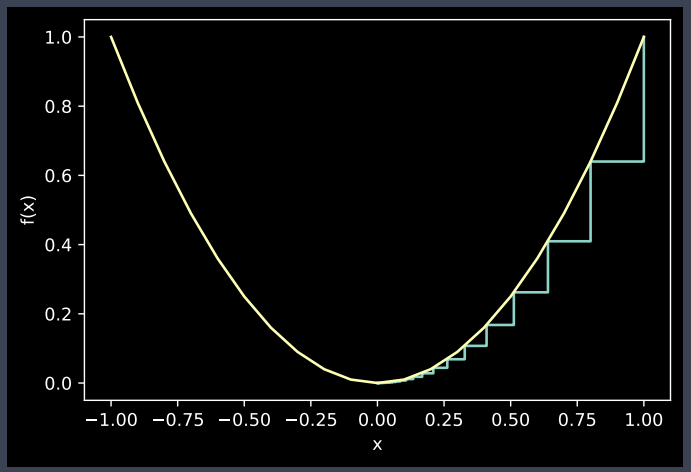
* Function decreases at every iteration, decrease gets smaller over time. Input x also decreases from 1 towards 0 (the optimum value).
* Hard to see small values of function, helps to plot on a log scale:



* On log-scale the function value is decreasing linearly towards zero i.e. $\log f(x_t) = mt + c$ with $m < 0$, so $f(x_t) = e^{mt+c}$ decreases exponentially $\rightarrow$ *super fast!*. This is basically best-case performance

» Example

* Starting value $x0 = 1$, step size $alpha = 0.1$
* Can be helpful to plot $(x, f(x))$ pairs of values over time.



* Why do the steps in x get smaller over time? Recall derivative is $2x$ and minimum is at $x = 0$. Derivates don't always diminish as get close to optimum, more on this later.

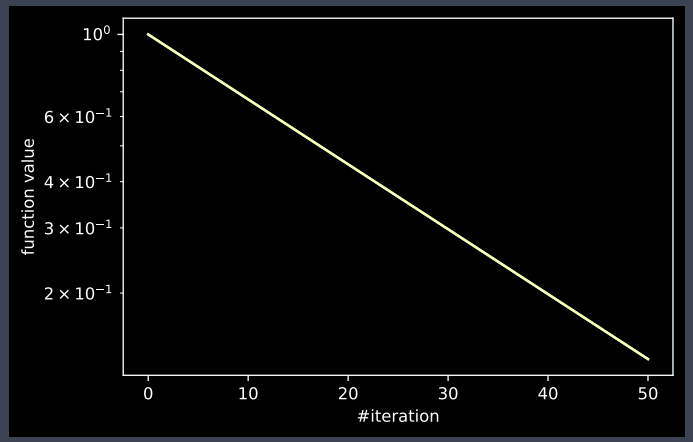
» Example

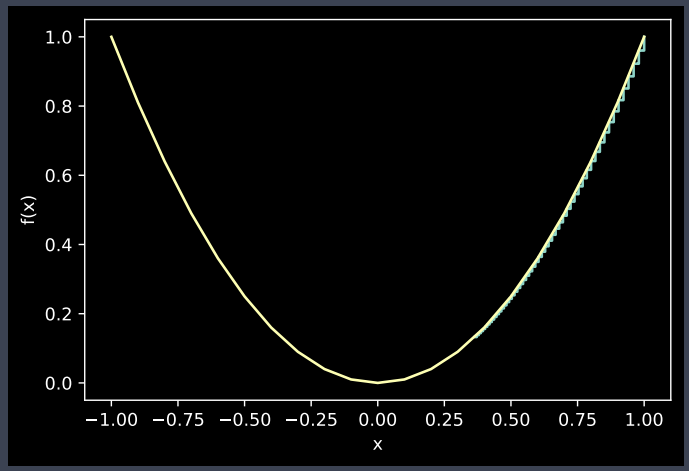
Plotting commands:

```
xx=np.arange(-1,1.1,0.1)
plt.plot(xx,fn.f(xx))
plt.xlabel('x'); plt.ylabel('f(x)')
plt.show()
plt.plot(F)
plt.xlabel('#iteration'); plt.ylabel('function value')
plt.show()
plt.semilogy(F)
plt.xlabel('#iteration'); plt.ylabel('function value')
plt.show()
plt.plot(X)
plt.xlabel('#iteration'); plt.ylabel('x')
plt.show()
plt.step(X,fn.f(X))
xx=np.arange(-1,1.1,0.1)
plt.plot(xx,fn.f(xx))
plt.xlabel('x'); plt.ylabel('f(x)')
plt.show()
```

» Example

* Starting value $x0 = 1$, step size $alpha = 0.01$

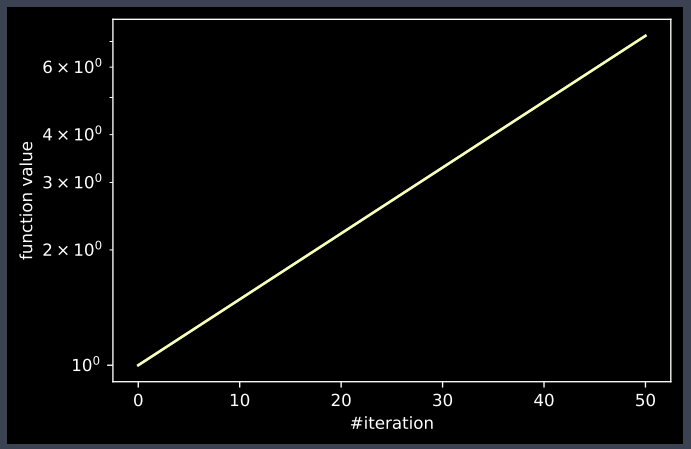


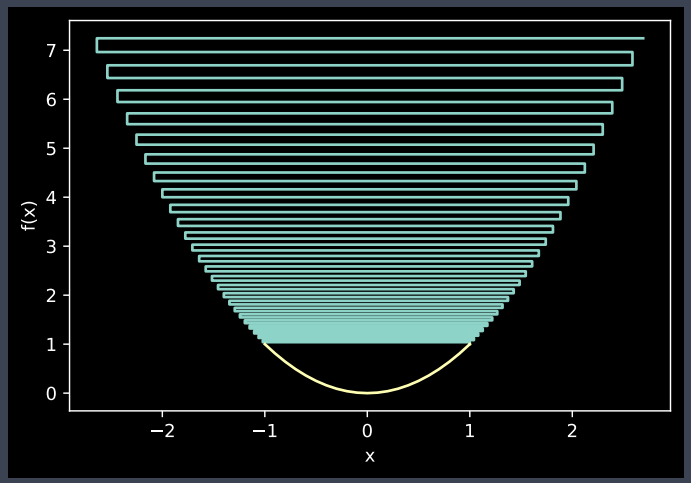


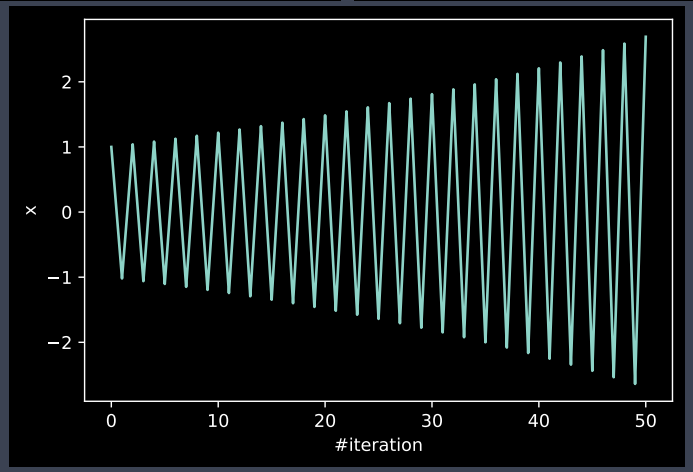
* Now rate of convergence is much smaller, and steps in x are smaller too.

» Example

* Starting value $x0 = 1$, step size $alpha = 1.01$



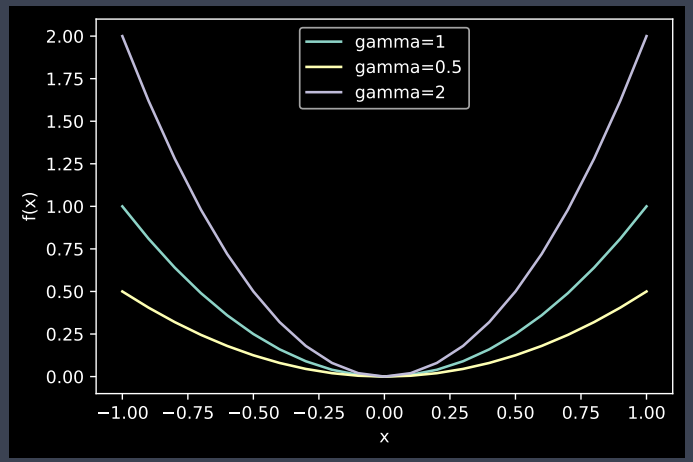




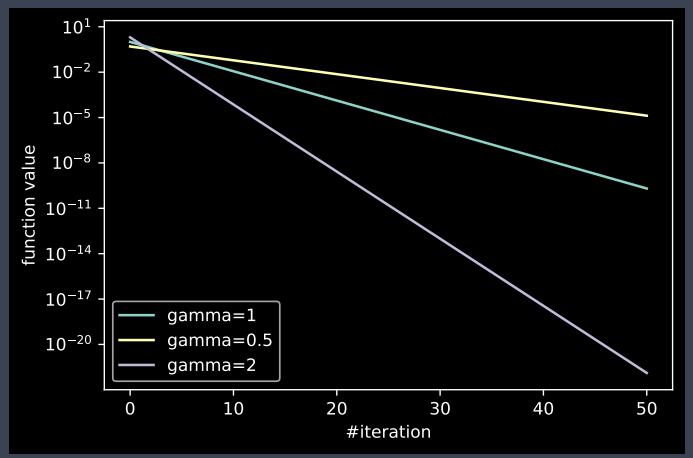
* Now no longer converges to minimum (in fact it diverges), steps in x are larger.

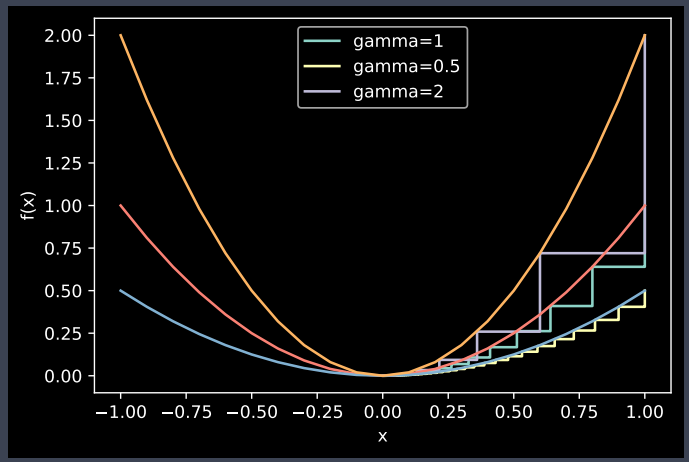
» Example

* What happens as function becomes more curved? Try $f(x) = \gamma x^2$ and vary γ.



* Starting value $x0 = 1$, step size $alpha = 0.1$

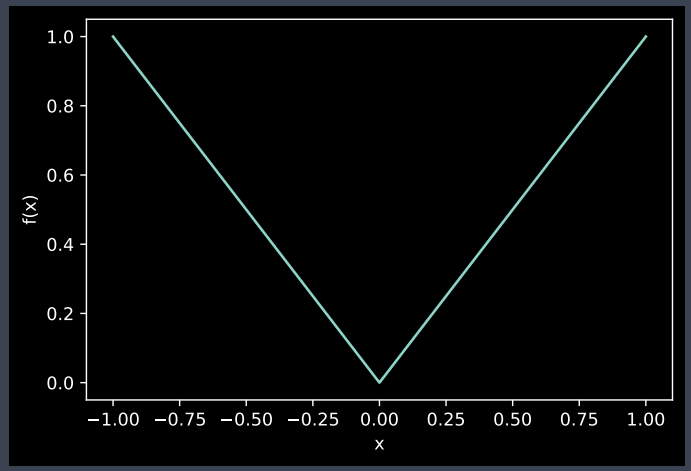




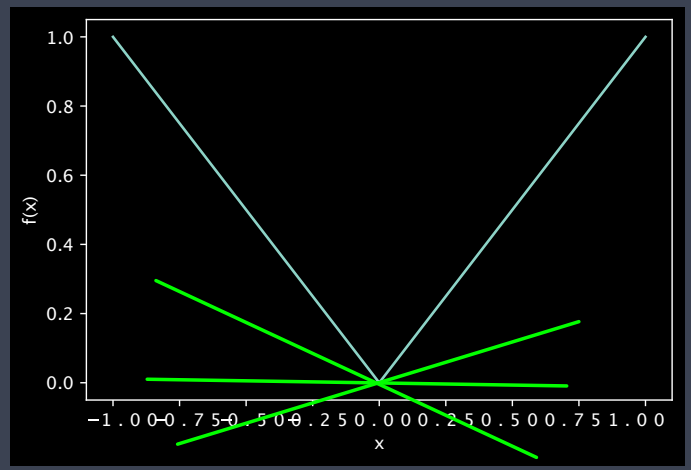
* Derivative is $2\gamma x$, so steps in x increase/decrease as γ increases/decreases $\rightarrow$ so we can expect to have to adjust step size α based on magnitude of derivative of $f(\cdot)$.

» Example 2

* Another example: $f(x) = |x|$. Minimum is 0 when $x = 0$.



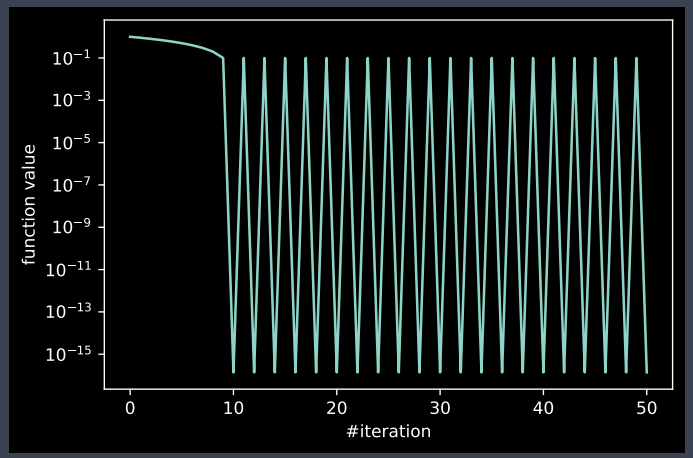
* Slope is constant, it doesn't decrease as get closer to minimum
* When $x > 0$ derivative of $f(\cdot)$ is $+1$, when $x < 0$ derivative is -1 i.e. derivative is $sign(x)$ for $x \neq 0$. When $x = 0$? No single line just touches the curve at that point ...

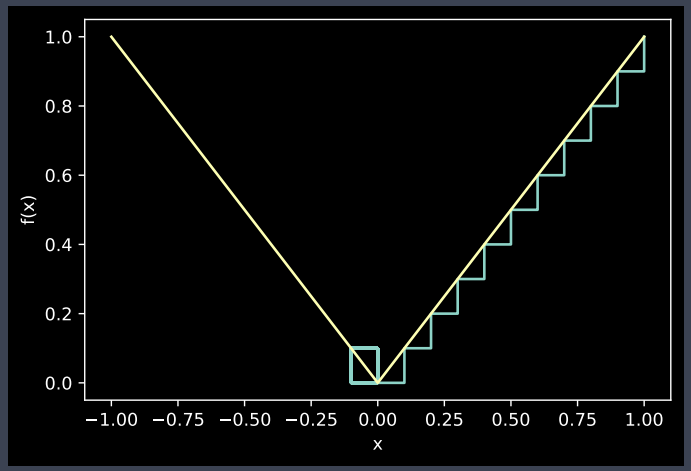


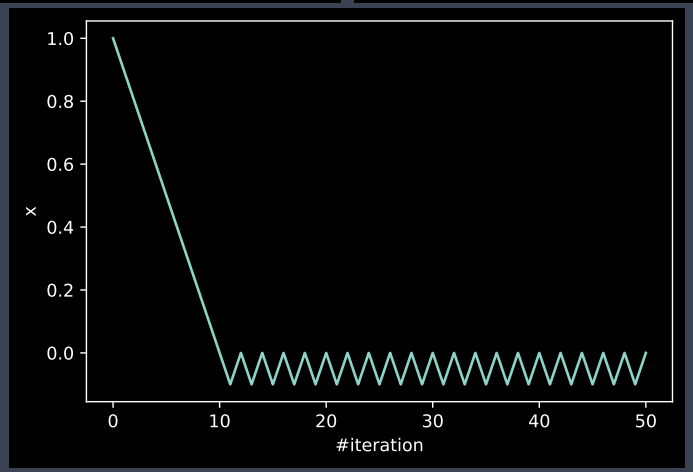
* Define derivative at $x = 0$ to be 0, will come back to this later.

» Example 2

* Starting value $x0 = 1$, step size $alpha = 0.1$



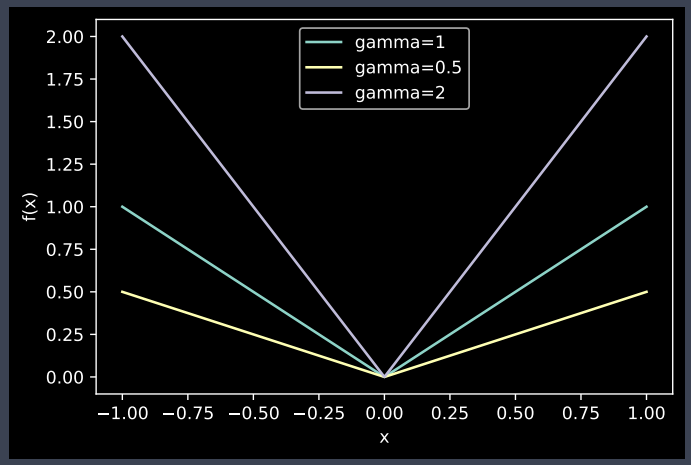




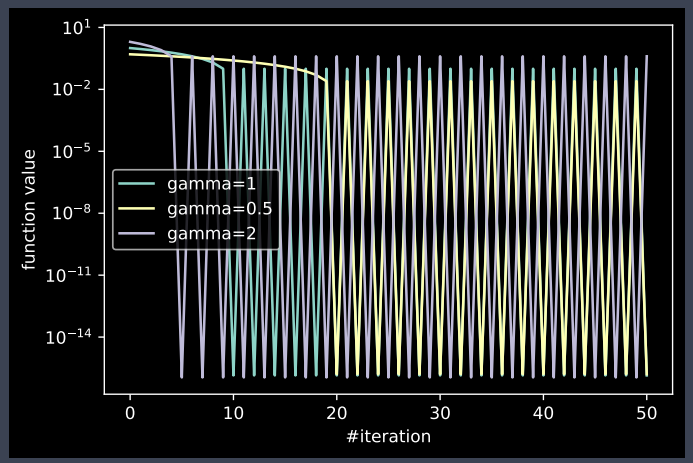
* Function decreases until it reaches vicinity of minimum, then oscillates forever - never converges to minimum.
* How might we try to fix that?

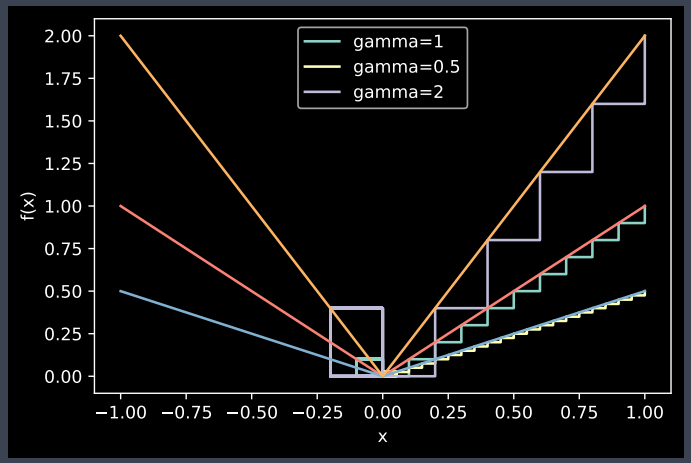
» Example 2

* As usual, changing magnitude of derivative affects steps. $f(x) = |\gamma x|$, derivative is $\gamma sign(x)$ for $x \neq 0$.



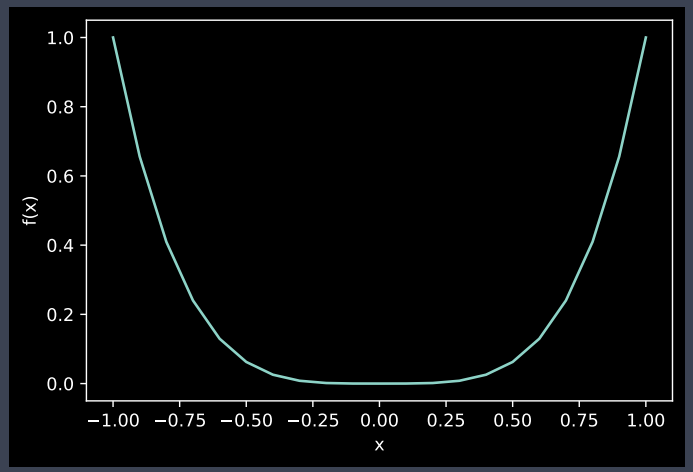
* Starting value $x0 = 1$, step size $alpha = 0.1$



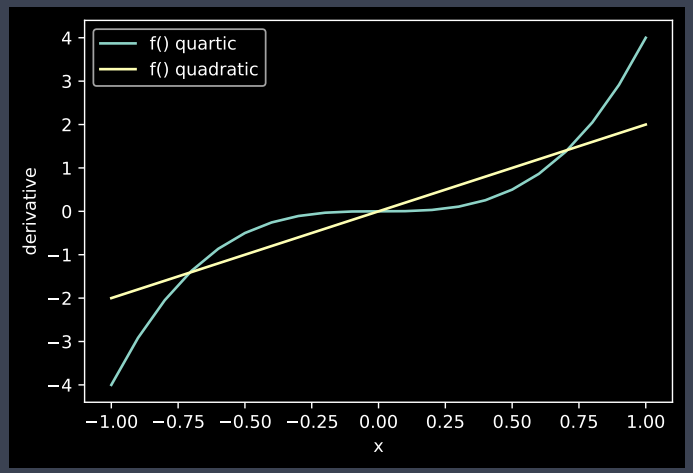


» Example 3

* Yet another example: $f(x) = x^4$. Minimum is 0 when $x = 0$.

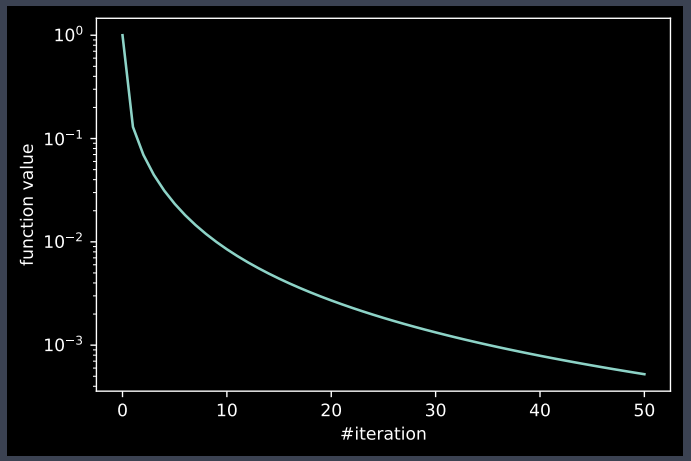


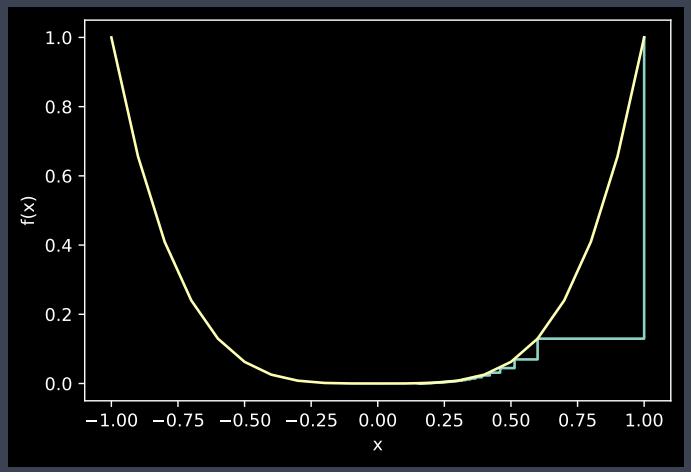
* Slope decreases more quickly than quadratic as get closer to minimum. Derivative of $f(x) = x^4$ is $3x^3$. Compare with derivative $2x$ when $f(x) = x^2$.

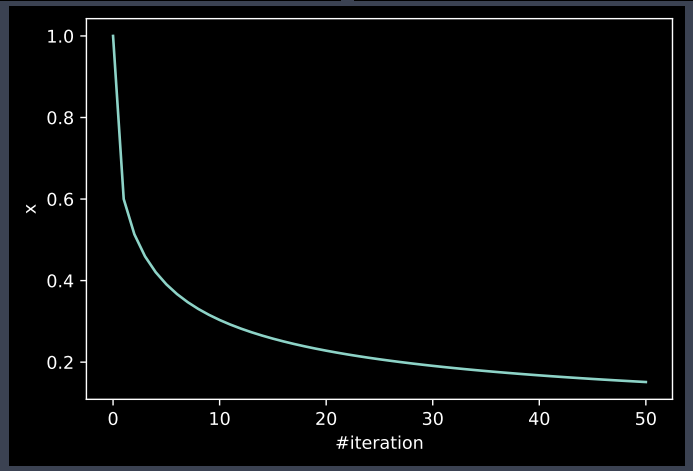


» Example 3

* Starting value $x0 = 1$, step size $alpha = 0.1$

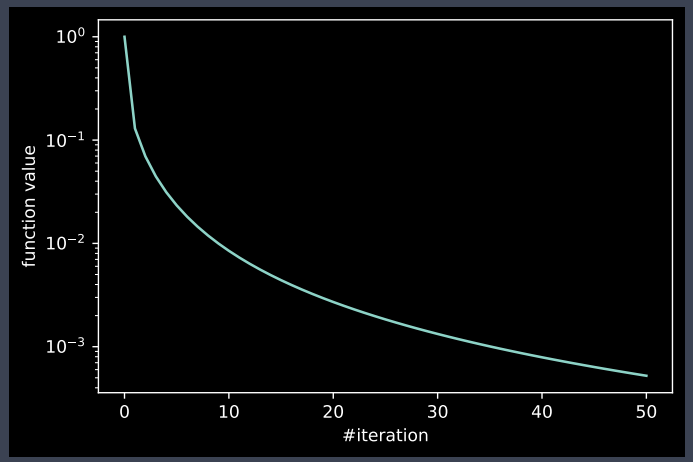


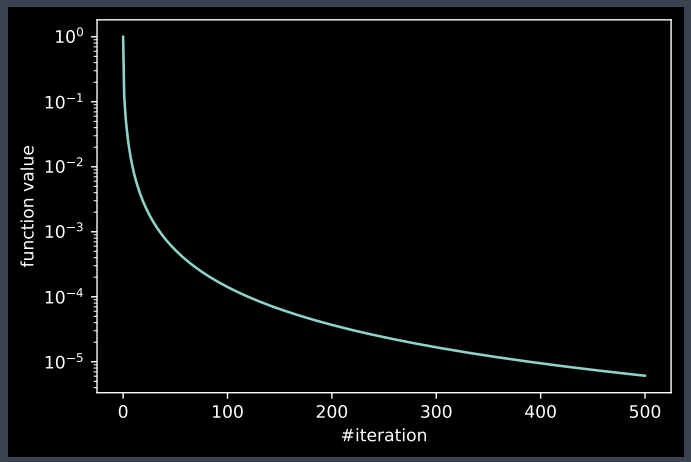




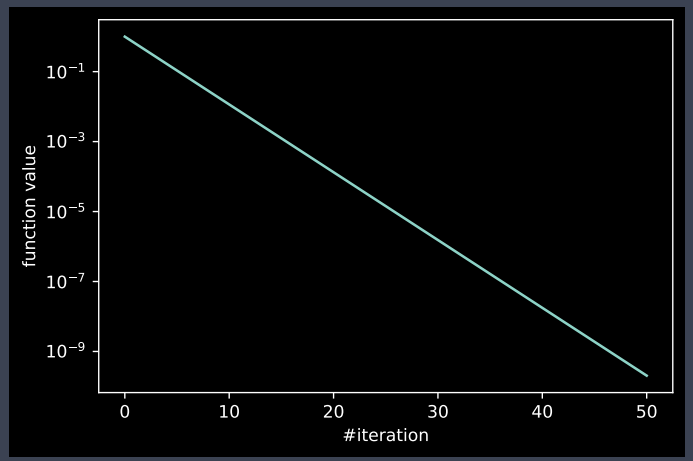
» Example 3

* Starting value $x0 = 1$, step size $alpha = 0.1$





* Observe that rate of decrease of $f(\cdot)$ gets slower and slower over time.
* Decrease of $\log f(x)$ is no longer linear, convergence rate is *much* slower than for quadratic function:



* How might we try to fix this?

» Summary

What have we discovered?

* No "one size fits all" choice of step size.
  * Scaling function e.g. from x^2 to $2x^2$ changes magnitude of derivative and so of steps taken
* Maybe need to change the step size over time
  * If derivative doesn't decrease as get close to minimum then can end up "bouncing around"
  * If derivative decreases too quickly as get close to minimum then can end up with v slow convergence
* Convergence rate can be v sensitive to properties of function being minimised
  * Quadratic function is "best" case, exponentially fast convergence (jargon: quadratic-like $\rightarrow$ *strongly convex*)
  * Kinks and flat areas in functions can greatly slow convergence (jargon: has kinks $\rightarrow$*non-smooth*)