CME 216, ME 343 - Winter 2021

Eric Darve, ICME

L-BFGS

BFGS can be expensive if the space is high-dimensional.

This cost can be reduced (with an approximation) using L-BFGS.

$$H_{k+1}y_k = s_k$$

$$H_{k+1} = (I - \rho_k s_k y_k^T) H_k (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T$$

Denote $V_k = I - \rho_k y_k s_k^T$.

$$H_{k+1} = V_k^T H_k V_k + \rho_k s_k s_k^T$$

Compute $H_{k+1}q$:

```
a[k] = rho[k]*s[k].T*q
q = q - a[k]*y[k] # Product with Vk
r = H[k]*q
b = rho[k]*y[k].T*r # Part of product with Vk^T
r = r - s[k]*b + s[k]*a[k]
```

$$H_{k+1} = (I - \rho_k s_k y_k^T) H_k (I - \rho_k y_k s_k^T) + \rho_k s_k s_k^T$$

Then repeat the same process to compute $H_k q$.

Unroll the recurrence m times.

$$H_{k+1} \rightarrow H_k \rightarrow \ldots \rightarrow H_{k-m+1}$$

```
for i = range(k, k-m, -1):
    a[i] = rho[i]*s[i].T*q
    q = q - a[i]*y[i]
r = H[k-m+1]*q
for i = range(k-m+1, k+1, 1):
    b = rho[i]*y[i].T*r
    r = r - s[i]*b + s[i]*a[i]
```

The catch is that we don't have `H[k-m+1]`.

In L-BFGS, we approximate this term by:

$$H_{k-m+1} \approx \gamma_{k+1} I$$

$$\gamma_{k+1} = \frac{s_k^T y_k}{y_k^T y_k}$$

$$\gamma_{k+1} = \frac{\boldsymbol{s}_k^T \boldsymbol{y}_k}{\boldsymbol{y}_k^T \boldsymbol{y}_k}$$

γ_{k+1} tries to approximate the norm of $\boldsymbol{H}_{k+1}$.

It's a crude approximation but since it is used to approximate a term far in the past, $\boldsymbol{H}_{k-m+1}$, it often works well.

The cost of L-BFGS is O(nm) if the Hessian has size n.

BFGS has cost O(n^2).

(Exact Trust Region has cost O(n^3).)

The cost of L-BFGS is much less than BFGS if $m \ll n$.

L-BFGS in scipy optimize.

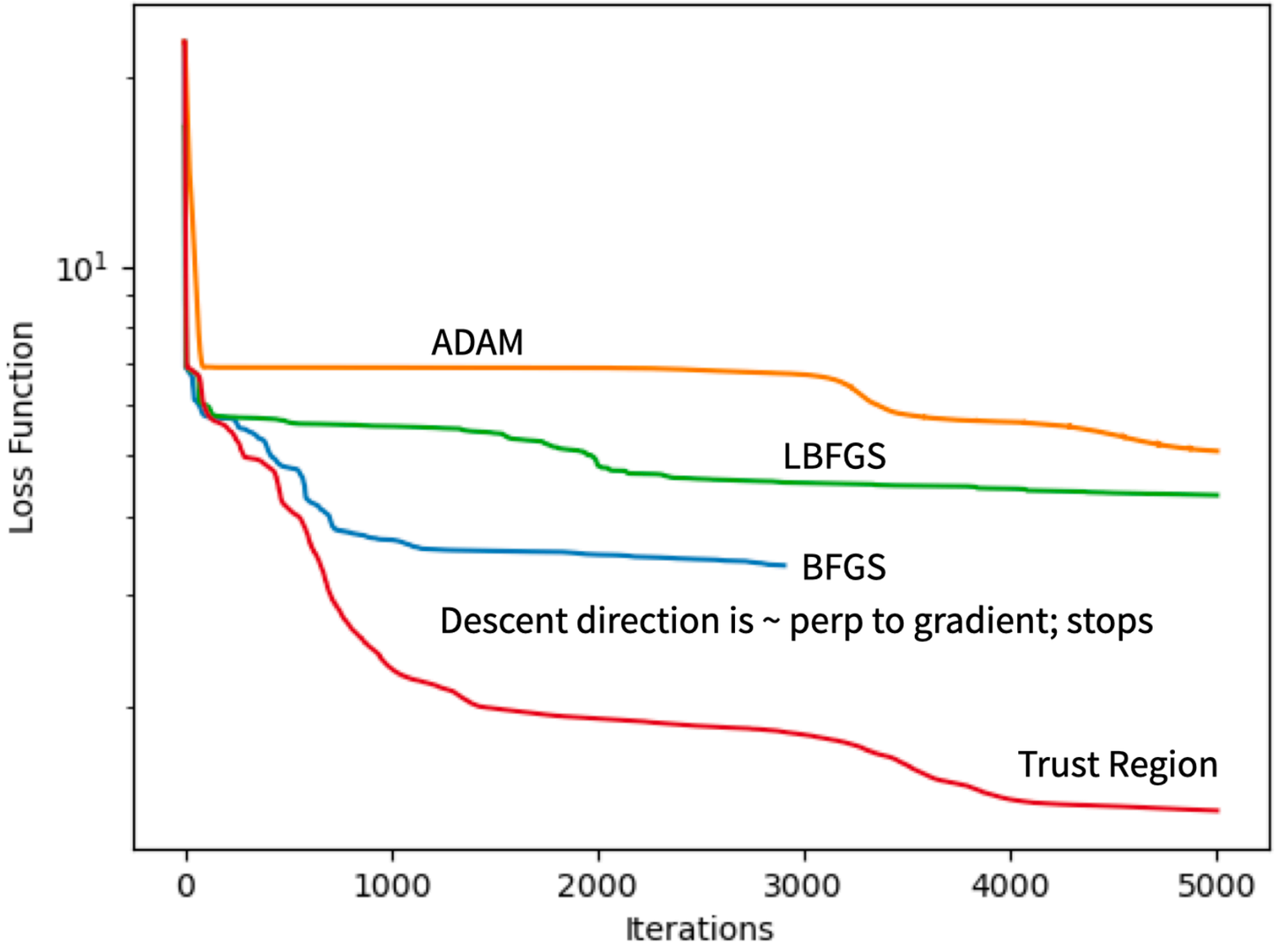
Let's see how this works in practice.

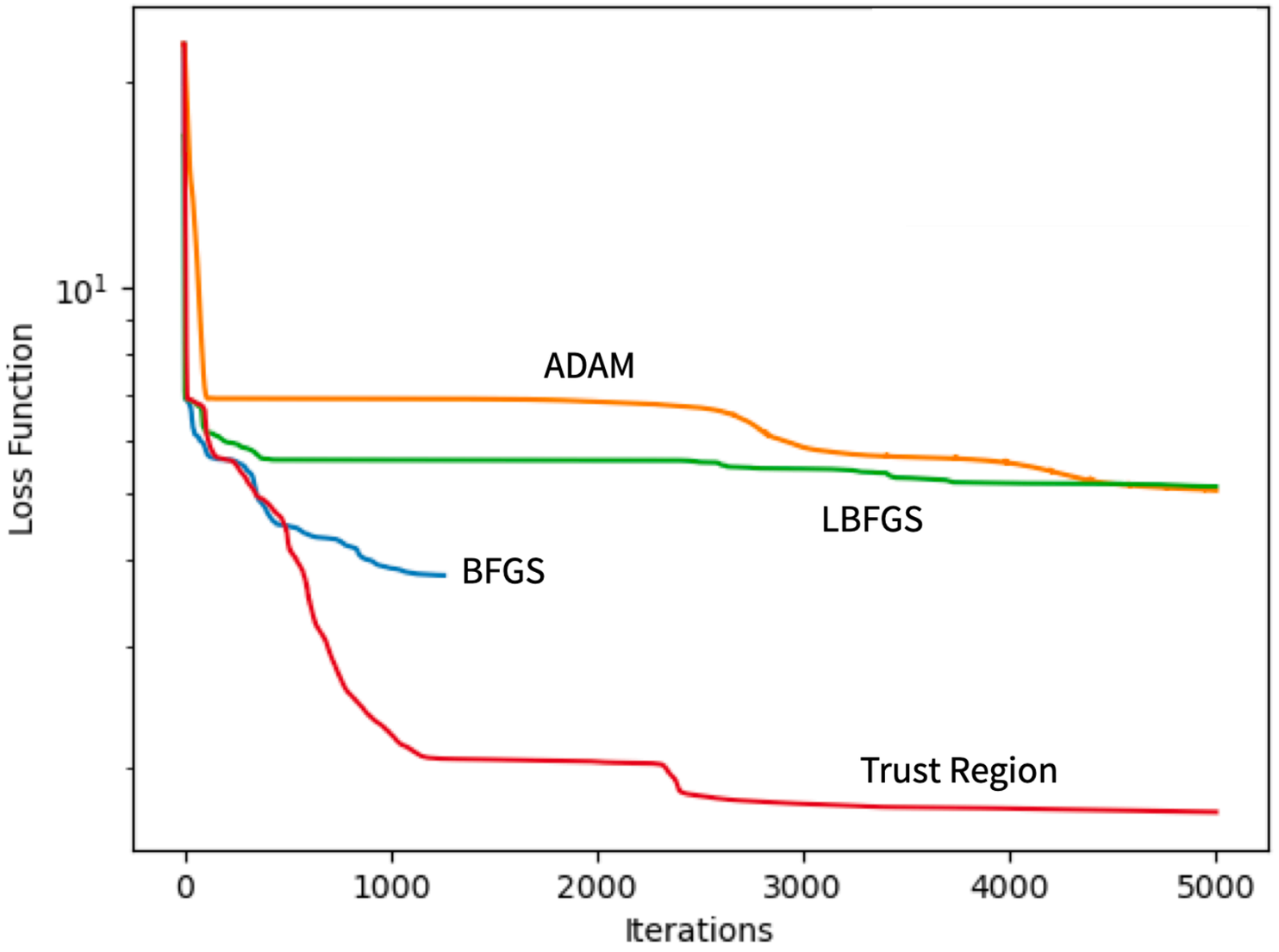
Physics-informed training task.

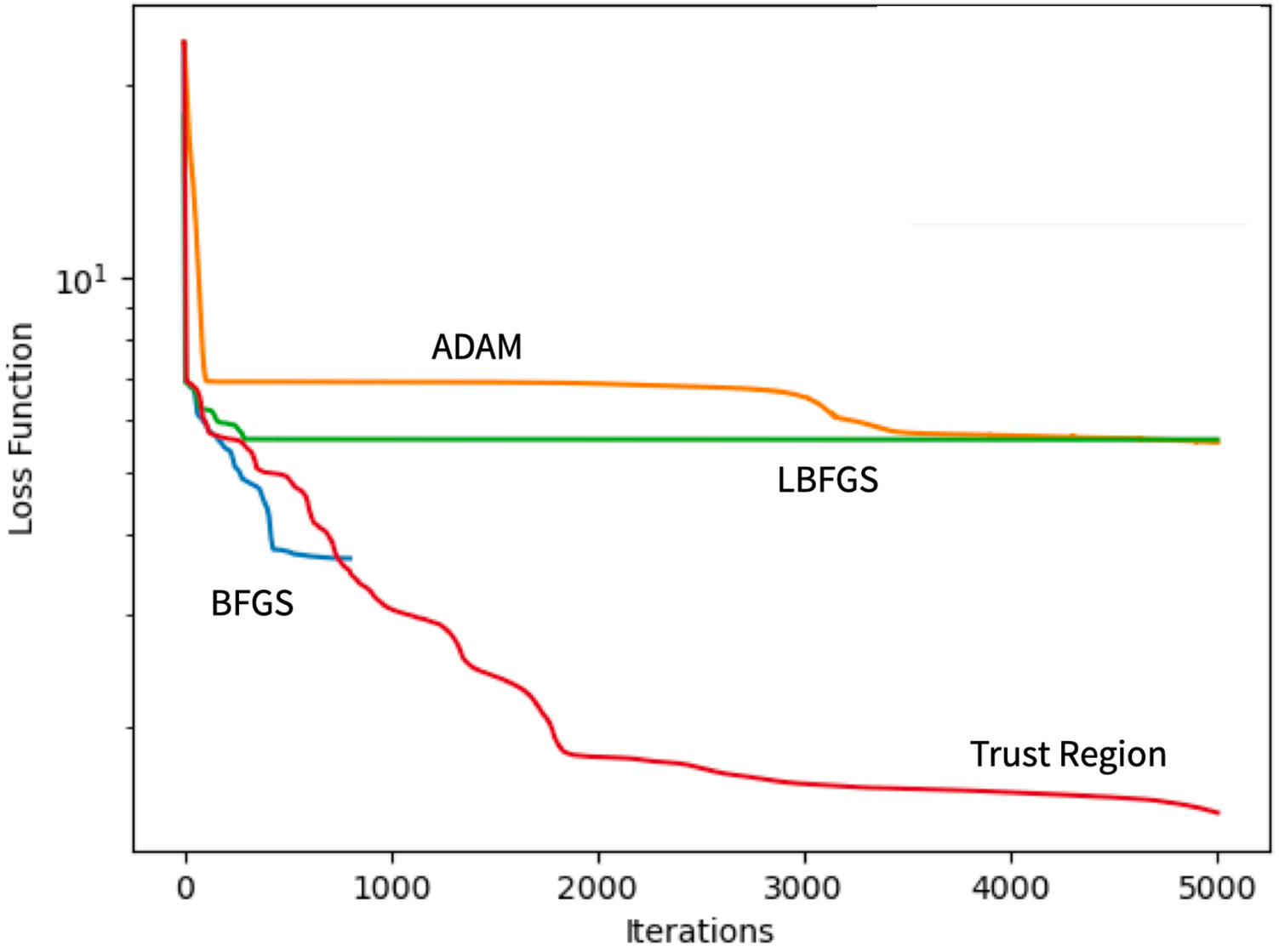
Poisson equation:

$$\nabla \cdot (\kappa_\theta(u) \nabla u) = f(x)$$

Learn $\kappa_\theta(u)$ using a DNN.


Loss Function
10^1
ADAM
LBFGS
BFGS
Descent direction is ~ perp to gradient; stops
Trust Region
0
1000
2000
3000
4000
5000
Iterations


Loss Function
10^1
ADAM
LBFGS
BFGS
Trust Region
0
1000
2000
3000
4000
5000
Iterations


Loss Function
10^1
ADAM
LBFGS
BFGS
Trust Region
0
1000
2000
3000
4000
5000
Iterations

But BFGS and L-BFGS require less flops.

Best method will depend on the application and many factors: accuracy, stiffness, cost of computing the Hessian, ...

For more information see Numerical Optimization, by Nocedal and Wright, Springer, 2nd edition