Lecture 6: Line search
A (more) global Newton's Method

COMP5930M Scientific Computation

Today

Line search

Line search algorithms

Implementation

Efficiency

Summary

A modified Newton algorithm with damping

We can view the Newton step as an n-d search for the solution $\mathbf{x}^*$

$$\begin{aligned} \mathbf{J}(\mathbf{x}_k)\delta &= -\mathbf{F}(\mathbf{x}_k) \\ \mathbf{x}_{k+1} &= \mathbf{x}_k + \lambda\delta \end{aligned}$$

- The vector δ is the current search direction
- We can then seek the optimal distance $\lambda > 0$ to move in that direction (avoiding long steps that cause divergence)
- **Note:** Not guaranteed that $\mathbf{F}(\mathbf{x}_k + \lambda\delta) = \mathbf{0}$ for any $\lambda > 0$.

A modified Newton algorithm with damping

$$\begin{aligned} \mathbf{J}(\mathbf{x}_k)\delta &= -\mathbf{F}(\mathbf{x}_k) \\ \mathbf{x}_{k+1} &= \mathbf{x}_k + \lambda\delta \end{aligned}$$

Criteria for choosing $\lambda > 0$:

- Should guarantee a reduction in $|\mathbf{F}(\mathbf{x}_{k+1})|^2 < |\mathbf{F}(\mathbf{x}_k)|^2$ (can't diverge, but not enough to guarantee convergence either)
- Should approximately minimise $|\mathbf{F}(\mathbf{x})|^2$ along the line segment $\mathbf{x}_k + \lambda\delta$

Computing λ

Two possible methods:

- A simple algorithm will begin with $\lambda = 1$ but reject any step that increases $|\mathbf{F}(\mathbf{x})|^2$
- A more complex algorithm computes the step length λ that minimises $|\mathbf{F}(\mathbf{x})|^2$ along the direction δ

1. Line-search by step-halving (Armijo rule)

- Find Newton search direction $\delta = -\left[\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right]^{-1} \mathbf{F}(\mathbf{x}_k)$
- Set $\lambda = 1$
- Test $|\mathbf{F}(\mathbf{x_k} + \lambda\delta)|^2 < |\mathbf{F}(\mathbf{x_k})|^2$
- Reject an increase, and repeat with $\lambda \to \frac{\lambda}{2}$

- Limit the number of rejects to guarantee completion

2. Line-search minimisation (quadratic interpolation)

- Using 3 points, construct a quadratic in λ to minimise the one-dimensional function:

$$\phi(\lambda) := |\mathbf{F}(\mathbf{x}_k + \lambda\delta_k)|^2.$$

  - Find an initial bracket $[\lambda_1, \lambda_2, \lambda_3]$ such that $\min\{\phi(\lambda_1), \phi(\lambda_3)\} > \phi(\lambda_2)$ and compute $\phi_i = \phi(\lambda_i)$.

2. Line-search minimisation (quadratic interpolation)

- Using 3 points, construct a quadratic in λ to minimise the one-dimensional function:

$$\phi(\lambda) := |\mathbf{F}(\mathbf{x}_k + \lambda\delta_k)|^2.$$

  - Find an initial bracket $[\lambda_1, \lambda_2, \lambda_3]$ such that $\min\{\phi(\lambda_1), \phi(\lambda_3)\} > \phi(\lambda_2)$ and compute $\phi_i = \phi(\lambda_i)$.
  - Define the quadratic polynomial $q_k(\lambda) := a\lambda^2 + b\lambda + c$ such that $q_k(\lambda_i) = \phi_i$ for $i = 1, 2, 3$

2. Line-search minimisation (quadratic interpolation)

- Using 3 points, construct a quadratic in λ to minimise the one-dimensional function:

$$\phi(\lambda) := |\mathbf{F}(\mathbf{x}_k + \lambda\delta_k)|^2.$$

  - Find an initial bracket $[\lambda_1, \lambda_2, \lambda_3]$ such that $\min\{\phi(\lambda_1), \phi(\lambda_3)\} > \phi(\lambda_2)$ and compute $\phi_i = \phi(\lambda_i)$.
  - Define the quadratic polynomial $q_k(\lambda) := a\lambda^2 + b\lambda + c$ such that $q_k(\lambda_i) = \phi_i$ for $i = 1, 2, 3$
  - Find the minimiser of q_k, $\lambda^* \in [\lambda_1, \lambda_3]$. The condition on the bracket guarantees that such a minimiser exists.
  - Set $x_{k+1} = x_k + \lambda^*\delta$ and go to next Newton step.

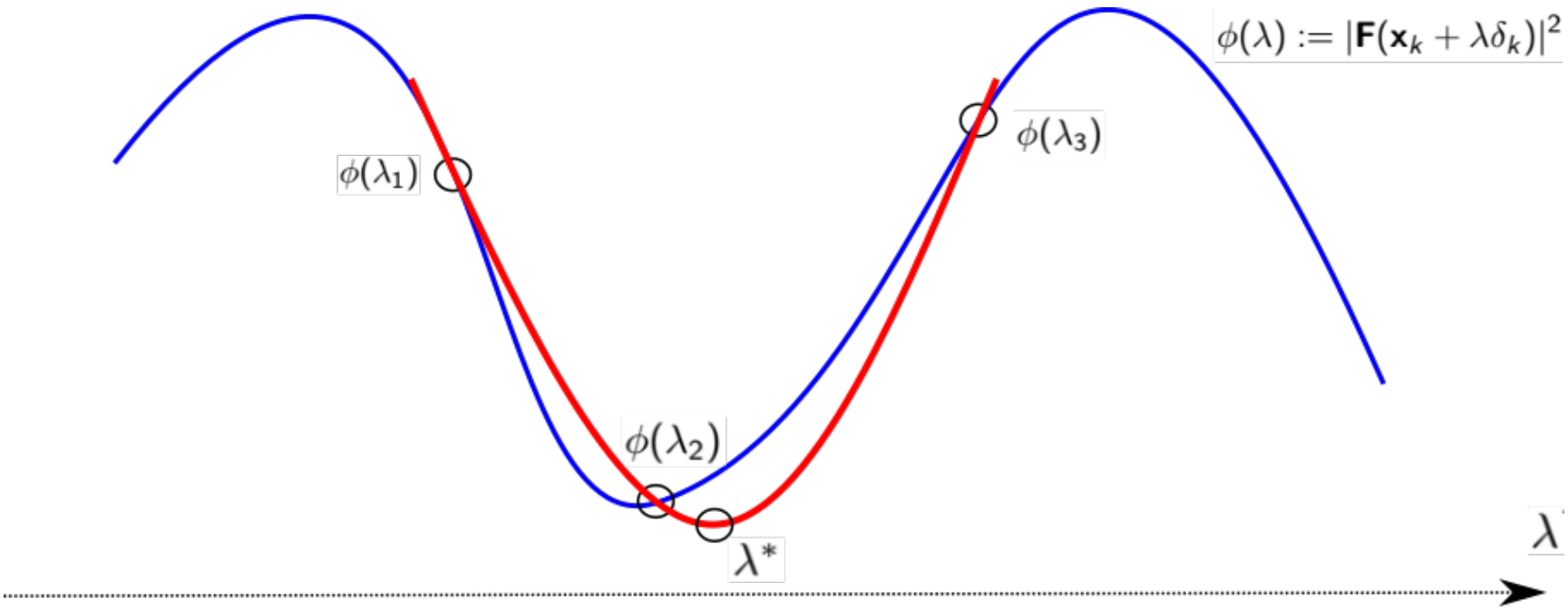
2. Line-search minimisation (quadratic interpolation)

- Using 3 points, construct a quadratic in λ to minimise the one-dimensional function:

$$\phi(\lambda) := |\mathbf{F}(\mathbf{x}_k + \lambda\delta_k)|^2.$$

  - Find an initial bracket $[\lambda_1, \lambda_2, \lambda_3]$ such that $\min\{\phi(\lambda_1), \phi(\lambda_3)\} > \phi(\lambda_2)$ and compute $\phi_i = \phi(\lambda_i)$.
  - Define the quadratic polynomial $q_k(\lambda) := a\lambda^2 + b\lambda + c$ such that $q_k(\lambda_i) = \phi_i$ for $i = 1, 2, 3$
  - Find the minimiser of q_k, $\lambda^* \in [\lambda_1, \lambda_3]$. The condition on the bracket guarantees that such a minimiser exists.
  - Set $x_{k+1} = x_k + \lambda^*\delta$ and go to next Newton step.

Illustration of quadratic interpolation



2. Line-search minimisation

- **Note:** It is not guaranteed the approximate line search with a quadratic results in a descent step $\phi(\mathbf{x}_{k+1}) < \phi(\mathbf{x}_k)$.
- We need to require a bit more to ensure convergence
- **Armijo condition:** Step length λ should satisfy

$$\phi(\mathbf{x}_k + \lambda\delta) \leq (1 - C\,\lambda\,)\phi(\mathbf{x}_k)$$

for some $0 < C < 1$ fixed (e.g. $C = 0.001$). This condition guarantees convergence to a **local minimum**.

2. Line-search minimisation (robust implementation)

- Using 3 points, construct a quadratic in λ and $|\mathbf{F}(\mathbf{x})|^2$ to minimise the one-dimensional function:

$$\phi(\lambda) := |\mathbf{F}(\mathbf{x}_k + \lambda\delta_k)|^2.$$

  - Find an initial bracket $[\lambda_1, \lambda_2, \lambda_3]$ such that $\min\{\phi(\lambda_1), \phi(\lambda_3)\} > \phi(\lambda_2)$ and compute $\phi_i = \phi(\lambda_i)$.
  - Define the quadratic polynomial $q_k(\lambda) := a\lambda^2 + b\lambda + c$ such that $q_k(\lambda_i) = \phi_i$ for $i = 1, 2, 3$
  - Find the minimiser of q_k, $\lambda^* \in [\lambda_1, \lambda_3]$. The condition on the bracket guarantees that such a minimiser exists.

2. Line-search minimisation (robust implementation)

Using λ^*, a new bracket is chosen:

- If $\lambda^* > \lambda_2$ and $\phi(\lambda^*) > \phi(\lambda_2)$ then the new bracket is $[\lambda_1, \lambda_2, \lambda^*]$.
- If $\lambda^* > \lambda_2$ and $\phi(\lambda^*) < \phi(\lambda_2)$ then the new bracket is $[\lambda_2, \lambda^*, \lambda_3]$.

2. Line-search minimisation (robust implementation)

Using λ^*, a new bracket is chosen:

- If $\lambda^* > \lambda_2$ and $\phi(\lambda^*) > \phi(\lambda_2)$ then the new bracket is $[\lambda_1, \lambda_2, \lambda^*]$.
- If $\lambda^* > \lambda_2$ and $\phi(\lambda^*) < \phi(\lambda_2)$ then the new bracket is $[\lambda_2, \lambda^*, \lambda_3]$.
- If $\lambda^* > \lambda_2$ and $\phi(\lambda^*) = \phi(\lambda_2)$, we can choose either one.

2. Line-search minimisation (robust implementation)

Using λ^*, a new bracket is chosen:

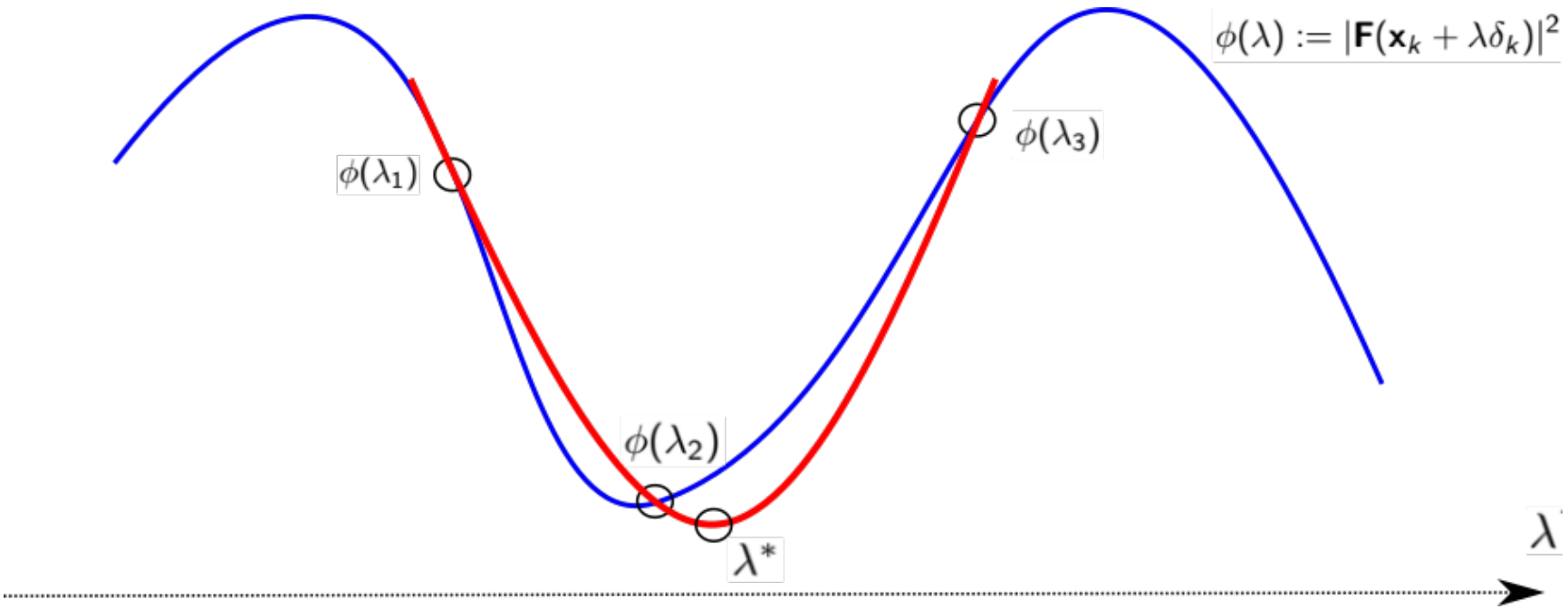
- If $\lambda^* > \lambda_2$ and $\phi(\lambda^*) > \phi(\lambda_2)$ then the new bracket is $[\lambda_1, \lambda_2, \lambda^*]$.
- If $\lambda^* > \lambda_2$ and $\phi(\lambda^*) < \phi(\lambda_2)$ then the new bracket is $[\lambda_2, \lambda^*, \lambda_3]$.
- If $\lambda^* > \lambda_2$ and $\phi(\lambda^*) = \phi(\lambda_2)$, we can choose either one.
- If $\lambda^* = \lambda_2$ the quadratic step has failed (no new point generated). We return to this case later.

2. Line-search minimisation (robust implementation)

Using λ^*, a new bracket is chosen:

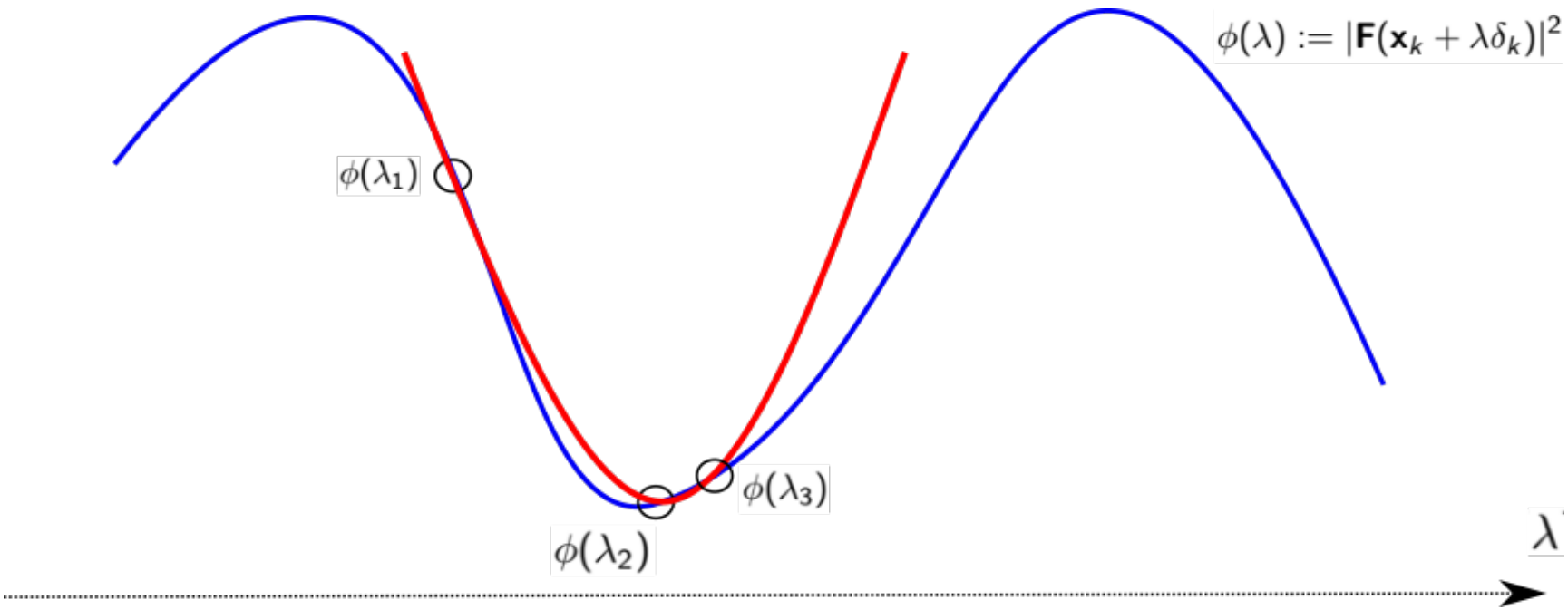
- If $\lambda^* > \lambda_2$ and $\phi(\lambda^*) > \phi(\lambda_2)$ then the new bracket is $[\lambda_1, \lambda_2, \lambda^*]$.
- If $\lambda^* > \lambda_2$ and $\phi(\lambda^*) < \phi(\lambda_2)$ then the new bracket is $[\lambda_2, \lambda^*, \lambda_3]$.
- If $\lambda^* > \lambda_2$ and $\phi(\lambda^*) = \phi(\lambda_2)$, we can choose either one.
- If $\lambda^* = \lambda_2$ the quadratic step has failed (no new point generated). We return to this case later.

Selection of new bracket



Here $\lambda^* > \lambda_2$ and $\phi(\lambda^*) > \phi(\lambda_2)$ so the new bracket is $[\lambda_1, \lambda_2, \lambda_*]$

Construction of new parabola



Now we find the next λ^* and iterate

2. Line-search minimisation (robust implementation)

Similar decision tree applies in the case $\lambda^* < \lambda_2$:

- If $\lambda^* < \lambda_2$ and $\phi(\lambda^*) > \phi(\lambda_2)$ then the new bracket is $[\lambda^*, \lambda_2, \lambda_3]$.
- If $\lambda^* < \lambda_2$ and $\phi(\lambda^*) < \phi(\lambda_2)$ then the new bracket is $[\lambda_1, \lambda^*, \lambda_2]$.
- If $\lambda^* < \lambda_2$ and $\phi(\lambda^*) = \phi(\lambda_2)$, we can choose either one.
- If $\lambda^* = \lambda_2$ the quadratic step has failed (no new point generated).

2. Line-search minimisation (robust implementation)

If the quadratic step failed, $\lambda^* = \lambda_2$, check if:

- Bracket is smaller than tolerance, $\lambda_3 - \lambda_1 <$TOL. If so, stop.
- Otherwise, perturb $\lambda_2 \to \lambda_2 \pm \varepsilon$ and retry quadratic step.

The quadratic step is iterated in the new bracket until:

$$\lambda_3 - \lambda_1 < TOL$$

or

$$\phi(\lambda_2) < TOL.$$

Newton's method

```
function [ x,f ] = newtonSysL( fnon, fjac, x0, tol, maxIt )

n = length(x0);

x = x0;                % initial point
f = feval(fnon,x);     % initial function values
normf = norm(f);
it = 0;

while (normf>tol) && (it<maxIt)

  J = feval( fjac, n, x, f, fnon ); % build Jacobian
  delta = -J\f;                     % solve linear system



  [ x,f,normf ] = lineSearch( fnon, x, normf, delta, 4 );



  it = it + 1;

end

end
```

Line search

```
function [ x,f,normf ] = lineSearch( fnon, x0, normf0, delta, maxStep )

step = 0;
lambda = 1;
x1 = x0 + delta;        % first update of x
f1 = feval(fnon,x1);    % new function values
normf1 = norm(f1);

while ( normf1 > normf0 )  && ( step < maxStep )
  step = step + 1;
  lambda = lambda/2;
  x1 = x0 + lambda*delta;  % update x
  f1 = feval(fnon,x1);
  normf1 = norm(f1);
end

x = x1;    % accept final value
f = f1;
normf = normf1;

end
```

Computational expense

For a single Newton iteration:

	Function calls	Algorithmic
Evaluate $\mathbf{J}$	n	n^2
Solve $\mathbf{J}\delta = -\mathbf{F}$	1	n^3
Update $\mathbf{x}_{k+1}$	$\approx$ 2-3	n

Line-search failure

- It is possible that the line search will not find a suitable point
- The algorithm extends the convergence of Newton's Method from purely local, but does not guarantee global convergence from any initial point $\mathbf{x}_0$
- A good initial point $\mathbf{x}_0$ is still critical

Comparison of Newton-based approaches

- Newton
  - Not robust in general for scalar case
  - Rarely converges in vector case
- Newton + bisection (scalar case only)
  - Dekker's method uses secant instead of Newton
  - Completely robust
- Newton + line-search
  - Better convergence but not guaranteed to be robust

Summary

- The practical Newton-type algorithm is composed of several non-trivial stages at each iteration
  - Solution of linear system to find Newton direction δ
  - Line-search to find optimal step length λ
- Each stage offers several algorithmic possibilities with performance and efficiency considerations
- The final choice is generally problem dependent