MA 580; Iterative Methods for Nonlinear Equations

C. T. Kelley
NC State University
tim_kelley@ncsu.edu

Version of November 4, 2016

Read Sections 6.1.1, 6.4, 8.1 of the Red book

NCSU, Fall 2016
Part VIIb: Newton-Krylov Methods and Global Convergence

Newton-Iterative Methods

Replace exact (or direct) solution of

$$\mathbf{F}'(\mathbf{x}_c)s = -\mathbf{F}(\mathbf{x}_c)$$

with an iterative method.
Terminate the linear (inner) iteration when the inexact Newton condition

$$\|\mathbf{F}'(\mathbf{x}_c)s + \mathbf{F}(\mathbf{x}_c)\| \leq \eta_c \|\mathbf{F}(\mathbf{x}_c)\|$$

holds.
η is called the forcing term.

Options

Examples: Newton-GMRES, Newton-MG, Newton-Krylov-Schwarz
Jacobian-vector product:

$$\mathbf{F}'(\mathbf{x})v \approx \frac{\mathbf{F}(\mathbf{x} + h\mathbf{v}) - \mathbf{F}(\mathbf{x})}{h}$$

where h is scaled to capture the low-order bits.

$$h = \|\mathbf{x}\|\sqrt{\epsilon_{mach}}/\|\mathbf{v}\|.$$

JFNK

- These methods are often called Jacobian-Free Newton Krylov (JFNK) methods.
- You will see the term JFNK all over computational science and engineering.
- Cost:
  - Nonlinear (Newton) iterations
  - Krylovs/Newton

Suppose, think H-equation, function/Jacobian cost $O(N^2)$ work

- Matrix factorization costs $N^3/3$ work.
- So if you get convergence (as we will see) in 6 Newtons and a total of 17 GMRES iterations **independently of N!!!**,
- your cost is $O(N^2)$, and

You've got a winner on your hands!

Convergence Theory

Theorem: Assume that $\eta_n \le \bar{\eta} < 1$, SA, and good data. Then the Newton iteration converges to x^* and

$$\|\mathbf{e}_{n+1}\| = O(\|\mathbf{e}_n\|^2 + \eta_n \|\mathbf{e}_n\|)$$

Proof: Cheating theorem.

Remarks

- If $\eta_n = O(\|\mathbf{F}(\mathbf{x}_n)\|)$ the convergence is quadratic.
- If $\eta_n \to 0$ the convergence is q-superlinear,

$$\|\mathbf{e}_{n+1}\|/\|\mathbf{e}_n\| \to 0.$$

- It is usually a poor idea to over solve in the inner iteration.

Integro-Differential Equation Example

Remember:

```
R_data=struct('D2',D2,'w',w,'x',x);

function f=ieq(u,R_data)
x=R_data.x; D2=R_data.D2; w=R_data.w;
f = D2*u + cos(x*u')*w;

function jac=ieqprime(u,R_data);
x=R_data.x; D2=R_data.D2; w=R_data.w;
jac=D2-diag(x)*sin(x*u')*diag(w);
```

Packaging things for the Jacobian-vector product

All we need to do is

- Write the Jacobian-vector product.
- Preconditon with $\mathbf{D}_2^{-1}$

So we will have to get orgainzed.

Example: Preconditioning Data

```
[L,U]=lu(D2);
PVEC=struct('L',L,'U',U);
```

Then pass the structure to the preconditioner

```
function pv = precondv(v,PVEC)
L=PVEC.L; U=PVEC.U; pv = U\(L\v);
```

Pass the Jacobian matrix directly to GMRES as in the linear case.

Communicating with GMRES

- Pass `MVEC` to `kl` as precomputed data for the preconditioner,
- and `jac` as precomputed data for the mat-vec.
- GMRES parameters:
    - `ltol = eta`
    - Commect `maxit` with η.

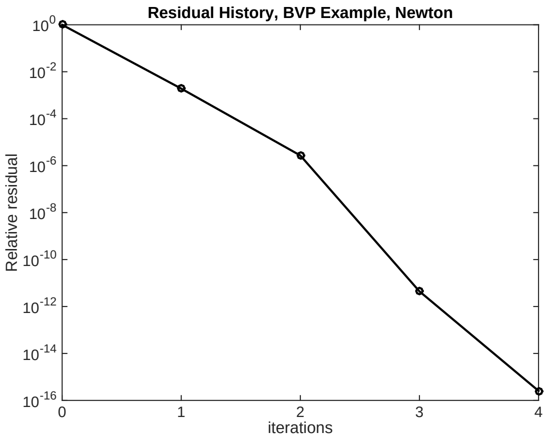
Call to `kl` to compute the step

Given

```
[L,U]=lu(D2); eta=.1;
res=ieq(u,R_data);
fp=ieqprime(u,R_data);
options=kl_optset('ltol',rtol,...
  'matvec_data',fp,'ptv',@precondv,...
   'p_side','right','p_data',PVEC);
[step, histl] = kl(u, -res, @mvec,options);
```

See `ieq_jfnk.m` on the Moodle page.

How did we do?


Residual History, BVP Example, Newton
Relative residual
iterations
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
10^{-12}
10^{-14}
10^{-16}
0
1
2
3
4

This is getting complicated

- What would a finite-difference Jacobian need?
  $\mathbf{u}$, $\mathbf{F}(\mathbf{u})$, data for $\mathbf{F}$, ...
- What if the function feeds data to the preconditioner?
- What if the preconditioner depends on the current iterate?
- What if GMRES is not your favorite linear solver?

Tools can manage this better than your writing a code every time.

Software

I use `knl.m` for the examples/homework.

- http://www4.ncsu.edu/~ctk/knl.html
- Link on Moodle page.
- You may also use `nsoli.m`
  http://www4.ncsu.edu/~ctk/newtony.html
- and you can write your own using `kl.m`, for example.

```
[sol, it_hist, ierr, x_hist] = knl(x,f, nloptions,static_data)
```

Input to `knl`

- `x`: initial iterate
- `f`: handle to the residual
- `nloptions`: options structure
- `static_data`: any precomputed data for `f`, the preconditioner, or the Jacobian-vector product.

The options are complex, because the algorithms are.

Output

- `sol`: solution
- `it_hist`: iteration history
- `ierr`: error flag
- `x_hist`: (optional) complete history of iteration

Options

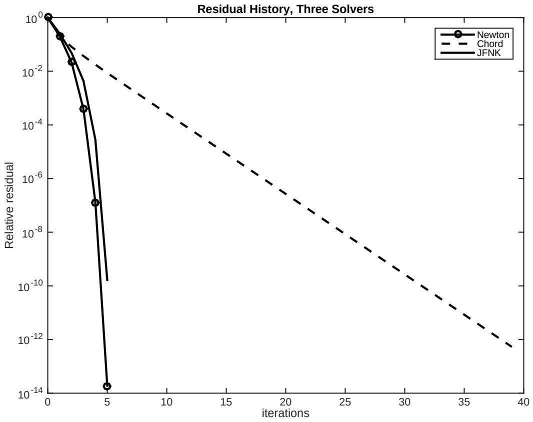
- `atol`, `rtol`, `maxit`, `maxitl`
- `etamax`: Forcing term control
- GMRES is the default linear solver
- FD is the default Jacobian-vector product

Here's the IEQ example ...

Calling knl

```
options=knl_optset('etamax',-.1,'atol',atol,...
  'rtol',rtol,'ptv',@precondv,...
   'p_side','right','p_data',PVEC);
[u,it_hist]=knl(u,@ieq,options,R_data);
histc=it_hist(:,1);
```

Residual History Plots: $H_0 \equiv 1$, $c = .975$



Newton vs Chord vs JFNK: timings on MacBook Air

- Residual histories are independent of N, but not of c.
- For $N = 5000$ and $c = .975$ the timings for a solve are
  - Newton: 14 secs, 5 iterations
  - Chord: 8 secs, 39 iterations
  - JFNK: .3 secs, 6 iterations
- For $N = 5000$ and $c = .5$ the timings for a solve are
  - Newton: 10 secs, 3 iterations
  - Chord: 3.5 secs, 6 iterations
  - JFNK: .2 secs, 6 iterations

Poor Data

Suppose you try to solve $\arctan(\mathbf{x}) = 0$ with Newton and $x_0 = 10$. The iterations are

$$10, -138, 2.9 \times 10^4, -1.5 \times 10^9, 9.9 \times 10^{17}.$$

What happened?

Line searches and the Armijo rule

Now we make a distinction between the Newton direction

$$\mathbf{d} = -\mathbf{F}'(\mathbf{x}_c)^{-1}\mathbf{F}(\mathbf{x}_c)$$

and the Newton step

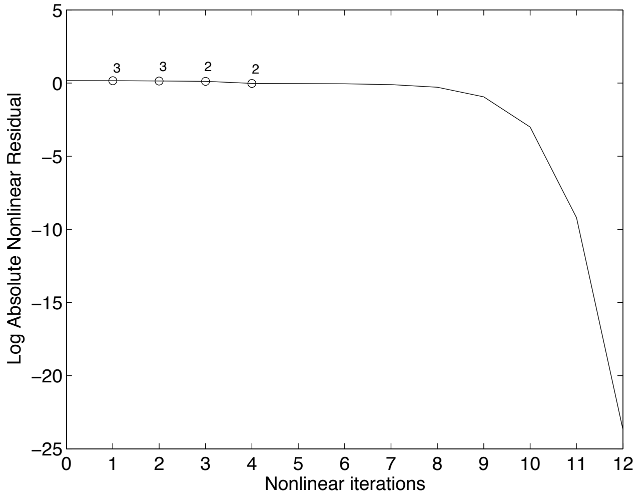
$$\mathbf{s} = \mathbf{x}_+ - \mathbf{x}_c$$

Simply put, we find the least $\lambda = 2^{-m}$ for $m = 0, 1, \dots$ so that

$$\|\mathbf{F}(\mathbf{x}_c + \lambda\mathbf{d})\| < \|\mathbf{F}(\mathbf{x}_c)\|$$

and use $\mathbf{s} = \lambda\mathbf{d}$. This process, a line search almost works.

Saving the ArcTan iteration



Sufficient Decrease and Termination

You need a little more to prove something. The sufficient decrease condition is

$$\|\mathbf{F}(\mathbf{x}_c + 2^{-m}\mathbf{d})\| < (1 - \alpha 2^{-m})\|\mathbf{F}(\mathbf{x}_c)\|,$$

Typical $\alpha = 10^{-4}$.
We will terminate when the nonlinear residual $\|F\|$ is small, i.e. when

$$\|\mathbf{F}(\mathbf{x}_n)\| \le \tau_a + \tau_r\|\mathbf{F}(\mathbf{x}_0)\|.$$

Newton-Iterative Algorithm

```
**nsolg**$(\mathbf{x}, F, \tau_a, \tau_r)$
    evaluate $\mathbf{F}(\mathbf{x})$; $\tau \leftarrow \tau_r|\mathbf{F}(\mathbf{x})| + \tau_a$.
    while $\|\mathbf{F}(\mathbf{x})\| > \tau$ do
        Find $d$ such that $\|\mathbf{F}'(\mathbf{x})\mathbf{d} + \mathbf{F}(\mathbf{x})\| \le \eta\|\mathbf{F}(\mathbf{x})\|$
        If no such $d$ can be found, terminate with failure.
        $\lambda = 1$
        while $\|\mathbf{F}(\mathbf{x} + \lambda\mathbf{d})\| > (1 - \alpha\lambda)\|\mathbf{F}(\mathbf{x})\|$ do
            $\lambda \leftarrow \sigma\lambda$ where $\sigma \in [1/10, 1/2]$
        end while
        $\mathbf{x} \leftarrow \mathbf{x} + \lambda\mathbf{d}$
    end while
```

Theory

Theorem: Suppose $\mathbf{F}$ is Lipschitz continuously differentiable, $\{\mathbf{x}_n\}$ is the inexact Newton-Armijo sequence, $0 < \eta_n < \bar{\eta} < 1$. The there are only three possibilities:

- $\{\mathbf{x}_n\}$ converges to a root $\mathbf{x}^*$ of $\mathbf{F}$ at which the standard assumptions hold, full steps $(\lambda = 1)$ are taken for n sufficiently large, and the local convergence theory holds.
- The sequence $\{\mathbf{x}_n\}$ is unbounded.
- The sequence $\{\mathbf{F}'(\mathbf{x}_n)^{-1}\}$ is unbounded.

A few examples

- $f(x) = e^x$; the Newton-Armijo sequence takes full steps and
$$x_+ = x_c - \frac{e^{x_c}}{e^{x_c}} = x_c - 1$$
So $x_n \to -\infty$.
- $f(x) = x^2 + 1$; You dont get full steps and $x_n \to 0$. What happened?

Bottom line: you can't solve a problem with no solution.

Choosing a Solver

The most important issues in selecting a solver are

- the size of the problem,
- the cost of evaluating $\mathbf{f}$ and $\mathbf{f}'$, and
- the way linear systems of equations will be solved.

The items in the list above are not independent.

Rough guidelines

- Small N and cheap $\mathbf{f}$; try direct methods and a forward difference Jacobian (but see the example at the end of this lecture for a different view).
- Large N or expensive $\mathbf{f}'$; try matrix-free Newton-Krylov solvers.
- Large N, very sparse $\mathbf{f}'$, only bad preconditioners; try sparse differencing for the Jacobian and use a direct method.