A history of Krylov Product Methods

A case of serendipity

34ste Woudschoten conferentie WSC

Peter Sonneveld

7 October 2009

Outline

- ‘Birth’ of the IDR-theorem? A side effect of an attempt to derive a more dimensional secant method.
- First practical use? An acceleration method for Gauss-Seidel iteration.
- Why was it abandoned so soon? The irresistible power of the CG-world.
  CGS, BiCGSTAB, BiCGSTAB(ℓ), ...
- Why is it reanimated now? Somebody mailed me with the question ‘What happened to IDR?’
- and finally: An explanation of the convergence!

1-dimensional secant method

Secant method is a quasi-Newton method for solving $f(x) = 0$. It constructs a sequence of approximations for $\boldsymbol{x}$: Solve

$$c_n f_n + d_n f_{n-1} = 0, \;\; c_n + d_n = 1$$

with $f_n = f(\boldsymbol{x}_n)$, etc. Calculate:

$$x_{n+1} = c_n x_n + d_n x_{n-1}$$

Let $e_n = x_n - x$, then

$$e_{n+1} \approx C e_n e_{n-1} \implies |e_{n+1}| \approx \widetilde{C} |e_n|^{1.618}$$

Secant cheaper than Newton since $1.618^2 \approx 2.618 > 2$

Poor Man's N-dim. Secant (*PMS*)

Is this possible in $\mathbb{R}^N$ for $N > 1$? Yes, but it is not a trivial thing.

Let

$$\sum_{k=0}^{N} c_{jk} \boldsymbol{f}_{j-k} = \mathbf{0}, \ \sum_{k=0}^{N} c_{jk} = 1, \ \boldsymbol{x}_{j+1} = \sum_{k=0}^{N} c_{jk} \boldsymbol{x}_{j-k}$$

At increasing j, system becomes ill-conditioned .

Safer variant: Replace only the 2 most recent vectors:

$[\boldsymbol{f}_j, \boldsymbol{f}_{j-1}] \Longrightarrow [\boldsymbol{f}_{j+1}, \boldsymbol{f}_j]$. This was tried out on a linear system $\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} = \mathbf{0}$, in order to watch asymptotic convergence behaviour.

What happened?

Serendipity moment: experiment 2006:

n	$\|\boldsymbol{f}_n\|$
0	2.2017e+00
1	2.6116e+00
2	1.3207e+00
3	6.7938e-01
4	8.1994e-01
5	8.6446e-01
6	8.4701e-01
7	7.8169e-01
8	9.9805e+00
9	2.6692e-01
10	4.6617e-02
11	7.9480e-03
12	3.4924e-04
13	1.1295e-04
14	4.4870e-14
15	8.0980e-16
16	1.1736e-16

Idealized secant method, digits=16, N=7

Why $\|\boldsymbol{f}_j\|$ drops at $j = 2N$?

It turned out that $\boldsymbol{f}_j$ were related by

$$\boldsymbol{f}_{j+1} = \boldsymbol{B}(c_{j0}\boldsymbol{f}_j + c_{j1}\boldsymbol{f}_{j-1}), \text{ with } c_{j0}\boldsymbol{f}_j + c_{j1}\boldsymbol{f}_{j-1} \perp \boldsymbol{p}, \; c_{j0} + c_{j1} = 1$$

$\boldsymbol{B}$ and $\boldsymbol{p}$ depend on the non-replaced vectors $\boldsymbol{f}_0, \boldsymbol{f}_1, \ldots, \boldsymbol{f}_{N-2}$. Hence they were fixed during the process.

The norm-drop at $j = 2N$ looks like a generic property of such a recurrence relation.

Experiment: Choose $\boldsymbol{B}$, $\boldsymbol{p}$, $\boldsymbol{f}_0$ randomly, $\boldsymbol{f}_1 = \boldsymbol{B}\boldsymbol{f}_0$.

Results from the random data process:

n	$\|\boldsymbol{f}_n\|$
0	1.2363e+00
1	1.5781e+00
2	8.1730e-01
3	5.2380e-01
4	7.4153e+00
5	3.3780e-01

n	$\|\boldsymbol{f}_n\|$
6	2.0971e-01
7	2.0858e-01
8	1.7339e-02
9	2.0244e-02
10	2.9124e-02
11	1.8132e-03

n	$\|\boldsymbol{f}_n\|$
12	1.6461e-04
13	7.2874e-06
14	1.1023e-18
15	9.6306e-19
16	6.0264e-19

drop-phenomenon, digits=16, N=7

Intermezzo: A finite Krylov solver! (1)

- The recursion between $\boldsymbol{f}_j$ in the simplified *PMS* process might be used for solving a linear system. Which system?
- Suppose $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$, then this recursion can be written as

$$\boldsymbol{s}_j = \boldsymbol{f}_j - \beta(\boldsymbol{f}_j - \boldsymbol{f}_{j-1}) \ \perp \boldsymbol{p}$$

$$\boldsymbol{f}_{j+1} - \boldsymbol{f}_j = (\boldsymbol{B} - \boldsymbol{I})\boldsymbol{s}_j - \beta(\boldsymbol{f}_j - \boldsymbol{f}_{j-1})$$

$$\boldsymbol{A}(\boldsymbol{x}_{j+1} - \boldsymbol{x}_j) = (\boldsymbol{I} - \boldsymbol{B})\boldsymbol{s}_j - \beta\boldsymbol{A}(\boldsymbol{x}_j - \boldsymbol{x}_{j-1})$$

- If $\boldsymbol{A} = \boldsymbol{I} - \boldsymbol{B}$, both sides have a left factor $\boldsymbol{A}$. Dividing out:

$$\boldsymbol{x}_{j+1} - \boldsymbol{x}_j = \boldsymbol{s}_j - \beta(\boldsymbol{x}_j - \boldsymbol{x}_{j-1})$$

Intermezzo: A finite Krylov solver! (2)

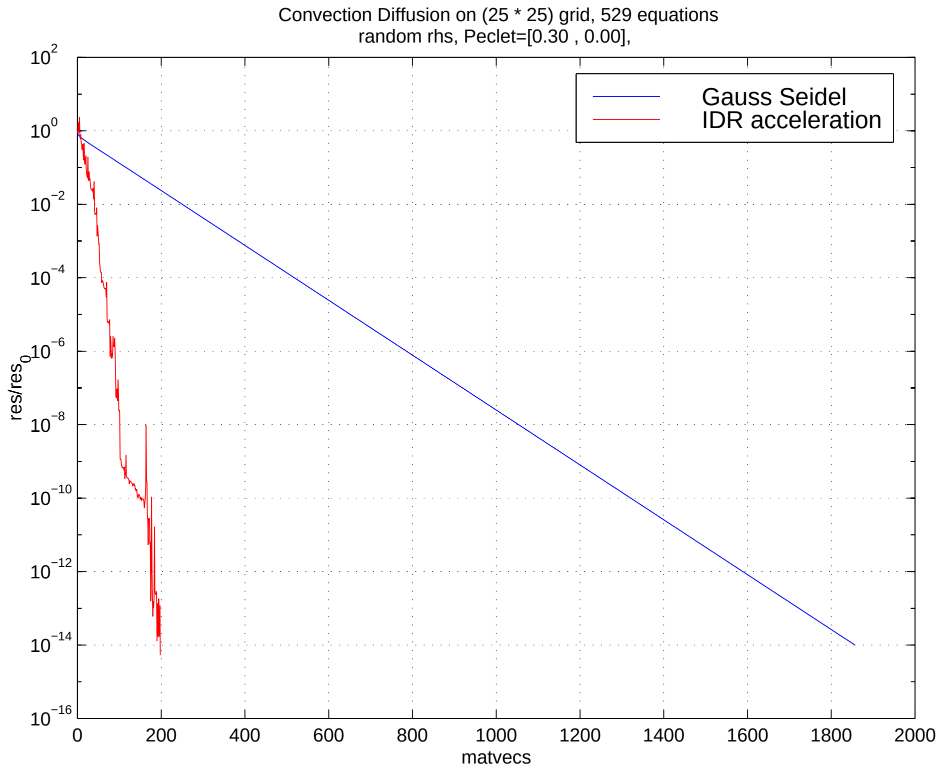
- Try out the process with $\boldsymbol{B}$ the Gauss Seidel matrix.
- Let $\boldsymbol{Ax} = \boldsymbol{b}$. Let $\boldsymbol{A} = \boldsymbol{L} + \boldsymbol{D} + \boldsymbol{U}$ with the obvious meaning.

$$(\boldsymbol{D}+\boldsymbol{L})\boldsymbol{x}_{j+1} = -\boldsymbol{U}\boldsymbol{x}_j + \boldsymbol{b} \Longrightarrow \boldsymbol{x}_{j+1} = -(\boldsymbol{D}+\boldsymbol{L})^{-1}(\boldsymbol{U}\boldsymbol{x}_j - \boldsymbol{b})$$

  So $\boldsymbol{B} = -(\boldsymbol{D}+\boldsymbol{L})^{-1}\boldsymbol{U} = \boldsymbol{I} - (\boldsymbol{D}+\boldsymbol{L})^{-1}\boldsymbol{A}$.

- Call the process ‘Accelerated Gauss Seidel’ (*AGS*).
- Apply to discrete Poisson equation on unit square, Dirichlet b.c.

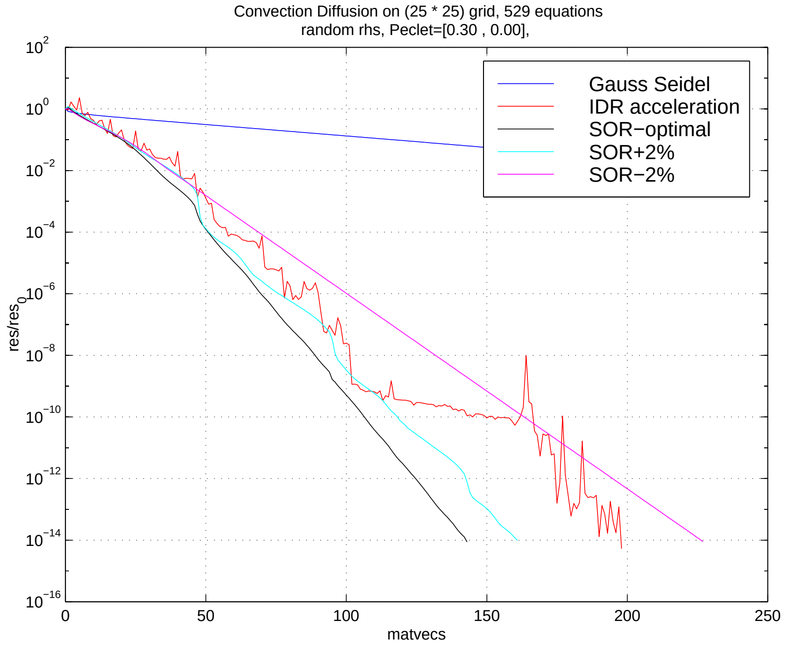
Results of AGS...



It works much faster than in 1058 steps!

Smashing! but how smashing??

Now compare to the real competitors ...



AGS can compete with SOR, ω_{opt} slightly to high. As problem size increases, SOR$(\omega_{\text{opt}} + 2\%)$ wins.

But determination of ω_{opt} is then increasingly difficult too!

Question: Why does it converge so fast?

and still Why is it finite?

Still, Why is this solver finite?!

- Consider the sequence $\{\boldsymbol{f}_0, \boldsymbol{f}_1, \ldots\}$, with $\boldsymbol{f}_1 = \boldsymbol{B}\boldsymbol{f}_0$, and

  $$\boldsymbol{f}_{j+1} = \boldsymbol{B}[\boldsymbol{f}_j - \beta_j(\boldsymbol{f}_j - \boldsymbol{f}_{j-1})]$$

  with β_j chosen such that $\boldsymbol{f}_j - \beta_j(\boldsymbol{f}_j - \boldsymbol{f}_{j-1}) \perp \boldsymbol{p}$.

- Then for $j \geq 2$, all $\boldsymbol{f}_j$ are in $\boldsymbol{B}(\boldsymbol{p}^\perp)$.

- Then for $j \geq 4$, $\boldsymbol{f}_{j-1} - \beta_{j-1}(\boldsymbol{f}_{j-1} - \boldsymbol{f}_{j-2})$ are in $\boldsymbol{p}^\perp \cap \boldsymbol{B}(\boldsymbol{p}^\perp)$. Hence for $j \geq 4$: $\boldsymbol{f}_j \in \boldsymbol{B}(\boldsymbol{p}^\perp \cap [\boldsymbol{B}(\boldsymbol{p}^\perp)]) \subset \boldsymbol{B}(\boldsymbol{p}^\perp)$.

- And so on and on and on.

- Define $\mathcal{G}_0 = \mathbb{R}^N$, and $\mathcal{G}_{j+1} = \boldsymbol{B}(\mathcal{G}_j \cap \boldsymbol{p}^\perp)$, then $\boldsymbol{f}_{i \geq 2j} \in \mathcal{G}_j$. Proposition: The $\mathcal{G}$- spaces form a *nest*: $\mathcal{G}_{j+1} \subset \mathcal{G}_j$.

Inductive proof of the nest-property

- First step: $\mathcal{G}_1 = \boldsymbol{B}(\mathcal{G}_0 \cap \boldsymbol{p}^\perp) \subset \mathcal{G}_0$.
- Assume $\mathcal{G}_j \subset \mathcal{G}_{j-1}$, then $(\mathcal{G}_j \cap \boldsymbol{p}^\perp) \subset (\mathcal{G}_{j-1} \cap \boldsymbol{p}^\perp)$.
- Therefore $\boldsymbol{B}(\mathcal{G}_j \cap \boldsymbol{p}^\perp) \subset \boldsymbol{B}(\mathcal{G}_{j-1} \cap \boldsymbol{p}^\perp)$, which is equivalent to $\mathcal{G}_{j+1} \subset \mathcal{G}_j$.
- Assume dim$(\mathcal{G}_{j+1}) <$ dim$(\mathcal{G}_j)$: $\mathcal{G}$-spaces are shrinking.
- Assume dim$(\mathcal{G}_{j+1}) =$ dim$(\mathcal{G}_j)$. Then $\mathcal{G}_{j+1} \equiv \mathcal{G}_j$, and $\mathcal{G}_j$ is an invariant subspace for $\boldsymbol{B}$. Also: $\mathcal{G}_j \subset \boldsymbol{p}^\perp$.
- For random $\boldsymbol{p}$, this will happen ‘almost never’
- This is the first (1976) Induced Dimension Reduction theorem.

IDR-theorem (1980)

Theorem 1 (IDR) *Let* $\boldsymbol{A}$ *be any matrix in* $\mathbb{R}^{N \times N}$*,*
let $\boldsymbol{v}_0$ *be any nonzero vector in* $\mathbb{R}^N$*,*
let $\mathcal{G}_0$ *be the full Krylov space* $\mathcal{K}^N(\boldsymbol{A}, \boldsymbol{v}_0)$*,*
let $\mathcal{S}$ *denote any (proper) subspace of* $\mathbb{R}^N$*, and let the sequence*
$\mathcal{G}_j$*,* $j = 1, 2, \ldots$ *be defined by*

$$\mathcal{G}_j = (\boldsymbol{I} - \omega_j \boldsymbol{A})(\mathcal{G}_{j-1} \cap \mathcal{S})$$

where ω_j *are nonzero numbers. Then*
i: $\mathcal{G}_j \subset \mathcal{G}_{j-1}$*, ii:* $M \leq N$ *exists such that*
$\mathcal{G}_j = \mathcal{G}_M, \; j = M + 1, M + 2, \ldots$

How to implement this

- Choose initial estimate $\boldsymbol{x}_0$, and a suitable vector $\boldsymbol{p} \in \mathbb{R}^N$. Let $\mathcal{S} = \boldsymbol{p}^\perp$.
- Start residual $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$, $\boldsymbol{x}_1 = \boldsymbol{x}_0 - \boldsymbol{r}_0$
  We have two residuals in $\mathcal{G}_0$.
- Assume $\boldsymbol{r}_{n-1}$ and $\boldsymbol{r}_n$ both in $\mathcal{G}_j$. Make $\boldsymbol{s}_n = \boldsymbol{r}_n - \beta(\boldsymbol{r}_n - \boldsymbol{r}_{n-1}) \perp \boldsymbol{p}$. Then $\boldsymbol{s}_n \in \mathcal{S} \cap \mathcal{G}_j$
- Then $\boldsymbol{r}_{n+1} = (\boldsymbol{I} - \omega_j \boldsymbol{A})\boldsymbol{s}_j$ is in $\mathcal{G}_{j+1}$
- Before calculating the first residual in $\mathcal{G}_{j+1}$, ω_n may be chosen free. Mostly to minimize $\|\boldsymbol{r}_{n+1}\|$

Primitive IDR-algorithm (1977)

$\boldsymbol{x}_0$ is initial guess; $\boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$.

Calculate ω_0 such that $\boldsymbol{r}_1 = \boldsymbol{r}_0 - \omega_0 \boldsymbol{A}\boldsymbol{r}_0$ in minimal in norm.

$d\boldsymbol{r} = \boldsymbol{r}_1 - \boldsymbol{r}_0$, $d\boldsymbol{x} = \omega_0 \boldsymbol{r}_0$, $\boldsymbol{x}_1 = \boldsymbol{x}_0 + d\boldsymbol{x}$.

For $j = 1, 2, \ldots$

Calculate β such that

$$d\boldsymbol{r} = \boldsymbol{r}_j - \boldsymbol{r}_{j-1},\ \ \boldsymbol{s}_j = \boldsymbol{r}_j - \beta d\boldsymbol{r}\ \perp \boldsymbol{p},\ \ \boldsymbol{t} = \boldsymbol{A}\boldsymbol{s}_j$$

If j is odd $\omega = \boldsymbol{t}^T\boldsymbol{s}/\boldsymbol{t}^T\boldsymbol{t}$; ($\|\boldsymbol{s} - \omega\boldsymbol{A}\boldsymbol{s}\|$ is made minimal)

$$d\boldsymbol{x} = \omega\boldsymbol{s}_j - \beta d\boldsymbol{x},\ \ d\boldsymbol{r} = -\omega\boldsymbol{t}_j - \beta d\boldsymbol{r}$$

$$\boldsymbol{r}_{j+1} = \boldsymbol{r}_j + d\boldsymbol{r},\ \ \boldsymbol{x}_{j+1} = \boldsymbol{x}_j + d\boldsymbol{x}$$

IDR-algorithm (1978)

```
$n = 0,\ \omega = 0,\ \boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$
$d\widehat{\boldsymbol{r}} = \boldsymbol{0},\ d\boldsymbol{y} = \boldsymbol{0},\ \gamma = 0$
while not convergent do
    $n = n + 1$
    $\boldsymbol{s} = \boldsymbol{r} + \gamma d\widehat{\boldsymbol{r}},\ \boldsymbol{t} = \boldsymbol{A}\boldsymbol{s}$
    if $(n = 1$ or $n$ even$)$: $\omega = (\boldsymbol{s}^T\boldsymbol{t})/(\boldsymbol{t}^T\boldsymbol{t})$
    $d\boldsymbol{x} = \gamma d\boldsymbol{y} + \omega\boldsymbol{s}$
    $d\boldsymbol{r} = \gamma d\widehat{\boldsymbol{r}} - \omega\boldsymbol{t}$
    $\boldsymbol{x} = \boldsymbol{x} + d\boldsymbol{x},\ \boldsymbol{r} = \boldsymbol{r} + d\boldsymbol{r}$
    if $n$ odd: $d\widehat{\boldsymbol{r}} = d\boldsymbol{r},\ d\boldsymbol{y} = d\boldsymbol{x}$
    $\gamma = -(\boldsymbol{p}^T\boldsymbol{r})/(\boldsymbol{p}^T d\widehat{\boldsymbol{r}})$
end
```

Convergence

- Why does IDR process converge?
- IDR is a Krylov subspace method
- Residuals satisfy $\boldsymbol{r}_n = \Phi_n(\boldsymbol{A})\boldsymbol{r}_0$, Φ_n is an n-th degree polynomial.
- Analyse those polynomials

The IDR polynomials.

- First element in $\mathcal{G}_j$, $\boldsymbol{r}_{2j}$, satisfies $\boldsymbol{r}_{2j} = \Phi_{2j}(\boldsymbol{A})\boldsymbol{r}_0$.
- For arbitrary $\boldsymbol{r} \in \mathcal{G}_j$, $\boldsymbol{r} = (1 - \omega_j \boldsymbol{A})\boldsymbol{r}'$, with $\boldsymbol{r}' \in \mathcal{G}_{j-1}$.
- Going down, finally : $\boldsymbol{r} = \Omega_j(\boldsymbol{A})\widetilde{\boldsymbol{r}}$, with $\widetilde{\boldsymbol{r}} \in \mathcal{G}_0$. Here $\Omega_j(t) = (1 - \omega_j t)(1 - \omega_{j-1} t). \cdots (1 - \omega_1 t)$.
- Hence Φ_{2j} is ‘divisible’ by Ω_j, $\Phi_{2j}(t) = \Omega_j(t)\phi_j(t)$
- From the intersections with $\boldsymbol{p}^{\perp}$ follows for $l < j$: $\boldsymbol{p}^T \Omega_l(\boldsymbol{A})\phi_j(\boldsymbol{A})\boldsymbol{r}_0 = 0$, so $\Omega_l(\boldsymbol{A}^T)\boldsymbol{p} \perp \phi_j(\boldsymbol{A})\boldsymbol{r}_0$.
  Hence ϕ_j is the j-th BiCG .- polynomial
- Obtained without calculating $\boldsymbol{A}^T$ products.

Why IDR was abandoned so soon?

- Relation with BiCG: Convergence analysis!
- Inner product $c = (\phi(\boldsymbol{A}^T)\boldsymbol{p})^T\psi(\boldsymbol{A})\boldsymbol{r}_0$ can be calculated as:

$$c = \boldsymbol{p}^T[\phi(\boldsymbol{A})\psi(\boldsymbol{A})\boldsymbol{r}_0]$$

Meaning possibly a transposefree BiGC algorithm?

Polynomial CG algorithm

Regular steps in (Bi-)CG algorithm:

$$\rho_n = \widetilde{\boldsymbol{r}}_n^T \boldsymbol{r}_n, \;\; \beta_n = \rho_n / \rho_{n-1}$$

$$\boldsymbol{p}_n = \boldsymbol{r}_n + \beta_n \boldsymbol{p}_{n-1}, \;\; \boldsymbol{q}_n = \boldsymbol{A}\boldsymbol{p}_n;$$

$$\widetilde{\boldsymbol{p}}_n = \widetilde{\boldsymbol{r}}_n + \beta_n \widetilde{\boldsymbol{p}}_{n-1}, \;\; \widetilde{\boldsymbol{q}}_n = \boldsymbol{A}^T \widetilde{\boldsymbol{p}}_n$$

$$\sigma_n = \widetilde{\boldsymbol{p}}_n^T \boldsymbol{q}_n, \;\; \alpha_n = \rho_n / \sigma_n$$

$$\boldsymbol{r}_{n+1} = \boldsymbol{r}_n - \alpha_n \boldsymbol{q}_n;$$

$$\widetilde{\boldsymbol{r}}_{n+1} = \widetilde{\boldsymbol{r}}_n - \alpha_n \widetilde{\boldsymbol{q}}_n$$

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \alpha_n \boldsymbol{p}_n$$

The polynomial relations

- The relevant vectors satisfy:

$$\boldsymbol{r}_n = \varphi_n(\boldsymbol{A})\boldsymbol{r}_0,\ \boldsymbol{p}_n = \psi_n(\boldsymbol{A})\boldsymbol{r}_0$$

$$\widetilde{\boldsymbol{r}}_n = \varphi_n(\boldsymbol{A}^T)\widetilde{\boldsymbol{r}}_0,\ \widetilde{\boldsymbol{p}}_n = \psi_n(\boldsymbol{A}^T)\widetilde{\boldsymbol{r}}_0$$

  where φ_n and ψ_n are polynomials of degree n.

- Define “inner product’ between polynomials:

$$\langle\phi_1, \phi_2\rangle = \boldsymbol{r}_0^T\phi_1(\boldsymbol{A})\phi_2(\boldsymbol{A})\boldsymbol{r}_0 = [\phi_1(\boldsymbol{A}^T)\boldsymbol{r}_0]^T\phi_2(\boldsymbol{A})\boldsymbol{r}_0$$

- then bi-orthogonality between $\boldsymbol{r}_n$ and $\widetilde{\boldsymbol{r}}_k$ corresponds to orthogonality of φ_n and φ_k.

CG for orthogonal polynomials

The coefficients and polynomials can also be calculated by

$$\rho_n = \langle \varphi_n, \varphi_n \rangle, \;\; \beta_n = \rho_n / \rho_{n-1}$$

$$\psi_n(t) = \varphi_n(t) + \beta_n \psi_{n-1}(t),$$

$$\sigma_n = \langle \psi_n, t\psi_n \rangle, \;\; \alpha_n = \rho_n / \sigma_n$$

$$\varphi_{n+1}(t) = \varphi_n(t) - \alpha_n t \psi_n(t)$$

This is (part of) CG-algorithm for orthogonal polynomials.

Algorithm for squared polynomials 1

Coefficients could as well have been calculated by

$$\rho_n = \langle 1, \varphi_n^2 \rangle, \;\; \sigma_n = \langle 1, t\psi_n^2 \rangle = \langle t, \psi_n^2 \rangle$$

But, then φ_n^2 and ψ_n^2 must be explicitly known. Assume that φ_n^2 and ψ_{n-1}^2 are explicitly known. Then

$$\psi_n^2(t) = \varphi_n^2(t) + 2\beta_n \varphi_n(t)\psi_{n-1}(t) + \beta_n^2 \psi_{n-1}^2(t)$$

$$\varphi_{n+1}^2(t) = \varphi_n^2(t) - 2\alpha_n t \varphi_n(t)\psi_n(t) + \alpha_n^2 t^2 \psi_n^2(t)$$

Extra required: $\varphi_n \psi_{n-1}$ and $\varphi_n \psi_n$, but these are dependent:

$$\varphi_n(t)\psi_n(t) = \varphi_n^2(t) + \beta_n \varphi_n(t)\psi_{n-1}(t)$$

Algorithm for squared polynomials 2

Descendant $\varphi_{n+1}\psi_n$ (of $\varphi_n\psi_{n-1}$):

$$\varphi_{n+1}(t)\psi_n(t) = \phi_n^2(t) + \beta_n\varphi_n(t)\psi_{n-1}(t) - \alpha_n t\psi_n^2(t)$$

Complete recursion:

Define $\Phi_n = \varphi_n^2, \quad \Theta_n = \varphi_n\psi_{n-1}, \quad \Psi_n = \psi_n^2$, then

$$\begin{array}{rcl} \Psi_n(t) & = & \Phi_n(t) + 2\beta_n\Theta_n(t) + \beta_n^2\Psi_{n-1}(t) \\ \Theta_{n+1}(t) & = & \Phi_n(t) + \beta_n\Theta_n(t) - \alpha_n t\Psi_n(t) \\ \Phi_{n+1}(t) & = & \Phi_n(t) - 2\alpha_n t[\Phi_n(t) + \beta_n\Theta_n(t)] + \alpha_n^2 t^2\Psi_n(t) \end{array}$$

where $\beta_n = \frac{\rho_n}{\rho_{n-1}}$, $\alpha_n = \frac{\rho_n}{\sigma_n}$, $\rho_n = \langle 1, \Phi_n\rangle$, and $\sigma_n = \langle t, \Psi_n\rangle$

Back to vectors

- $\widehat{\boldsymbol{r}}_n = \Phi_n(\boldsymbol{A})\boldsymbol{r}_0$, $\widehat{\boldsymbol{p}}_n = \Psi_n(\boldsymbol{A})\boldsymbol{r}_0$, and $\widehat{\boldsymbol{q}}_n = \Theta_n(\boldsymbol{A})\boldsymbol{r}_0$.
- Substitute $\boldsymbol{A}$ for t in the 'squared polynomial algorithm', and apply the obtained operators to $\boldsymbol{r}_0$. Then we get

$$\begin{array}{rcl} \widehat{\boldsymbol{p}}_n & = & \widehat{\boldsymbol{r}}_n + 2\beta_n\widehat{\boldsymbol{q}}_n + \beta_n^2\widehat{\boldsymbol{p}}_{n-1} \\ \widehat{\boldsymbol{q}}_{n+1} & = & \widehat{\boldsymbol{r}}_n + \beta_n\widehat{\boldsymbol{q}}_n - \alpha_n\boldsymbol{A}\widehat{\boldsymbol{p}}_n \\ \widehat{\boldsymbol{r}}_{n+1} & = & \widehat{\boldsymbol{r}}_n - 2\alpha_n\boldsymbol{A}[\widehat{\boldsymbol{r}}_n + \beta_n\widehat{\boldsymbol{q}}_n] + \alpha_n^2\boldsymbol{A}^2\widehat{\boldsymbol{p}}_n \end{array}$$

- $\widehat{\boldsymbol{r}}_{n+1} - \boldsymbol{r}_n$ is 'divisible by $\boldsymbol{A}$', so we can update the solution:

$$\widehat{\boldsymbol{x}}_{n+1} = \widehat{\boldsymbol{x}}_n + 2\alpha_n[\widehat{\boldsymbol{r}}_n + \beta_n\widehat{\boldsymbol{q}}_n] - \alpha_n^2\boldsymbol{A}\widehat{\boldsymbol{p}}_n$$

Heart of CGS algorithm

Careful translation of squared-polynomial algorithm to vectors:

$$\rho_n = \widetilde{\boldsymbol{r}}_0^T \widehat{\boldsymbol{r}}_n, \;\; \beta_n = \rho_n / \rho_{n-1}$$

$$\widehat{\boldsymbol{u}} = \widehat{\boldsymbol{r}}_n + \beta_n \widehat{\boldsymbol{q}}_n$$

$$\widehat{\boldsymbol{p}}_n = \widehat{\boldsymbol{u}} + \beta_n (\widehat{\boldsymbol{q}}_n + \beta_n \widehat{\boldsymbol{p}}_{n-1})$$

$$\widehat{\boldsymbol{v}} = \boldsymbol{A} \widehat{\boldsymbol{p}}_n, \;\; \sigma_n = \widetilde{\boldsymbol{r}}_0^T \widehat{\boldsymbol{v}}, \;\; \alpha_n = \rho_n / \sigma_n$$

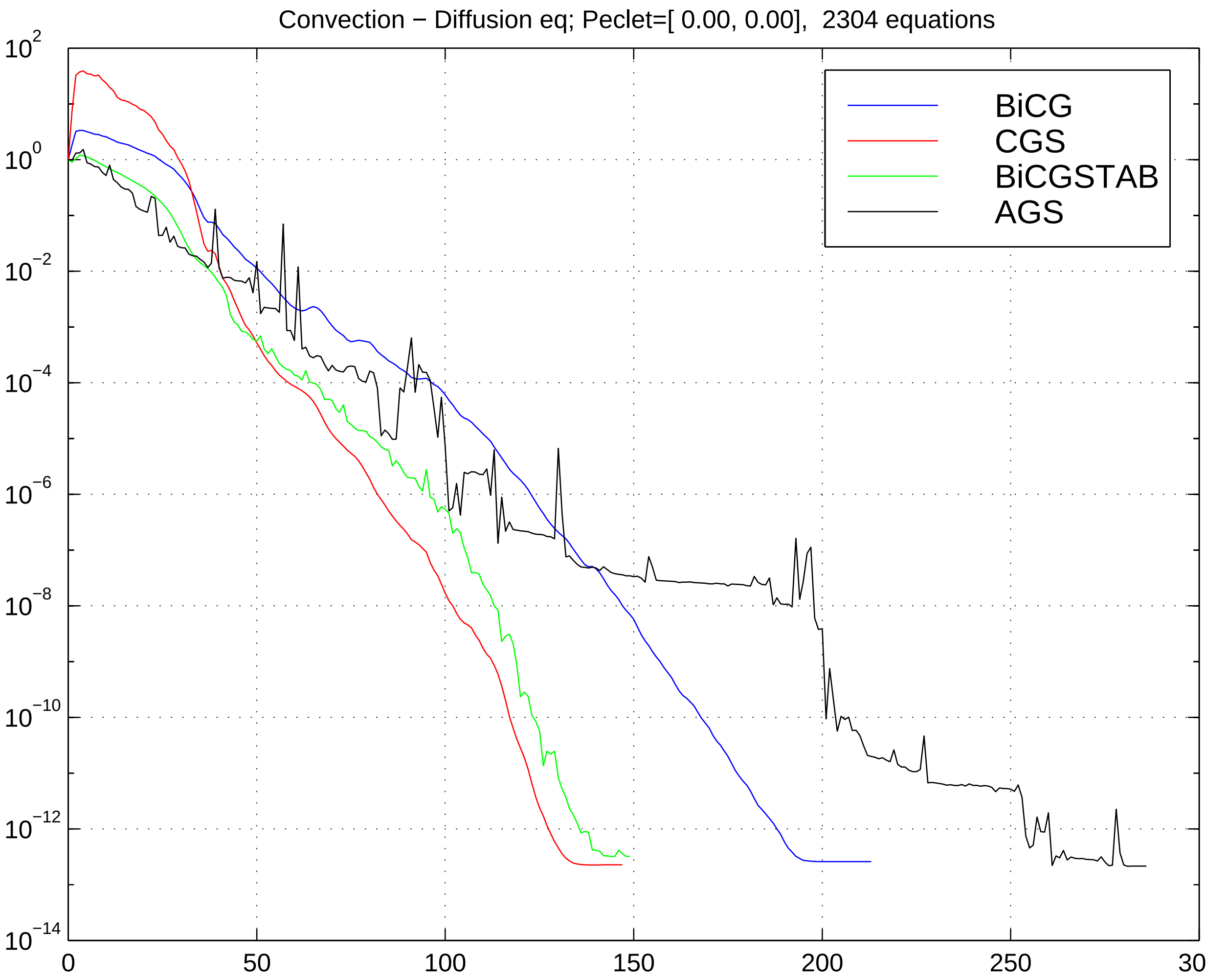
$$\widehat{\boldsymbol{q}}_{n+1} = \widehat{\boldsymbol{u}} - \alpha_n \widehat{\boldsymbol{v}}$$

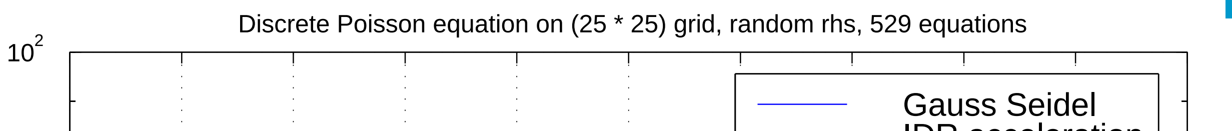
$$\widehat{\boldsymbol{v}} = \alpha_n (\widehat{\boldsymbol{u}} + \widehat{\boldsymbol{q}}_{n+1})$$

$$\widehat{\boldsymbol{r}}_{n+1} = \widehat{\boldsymbol{r}}_n - \boldsymbol{A} \widehat{\boldsymbol{v}}$$

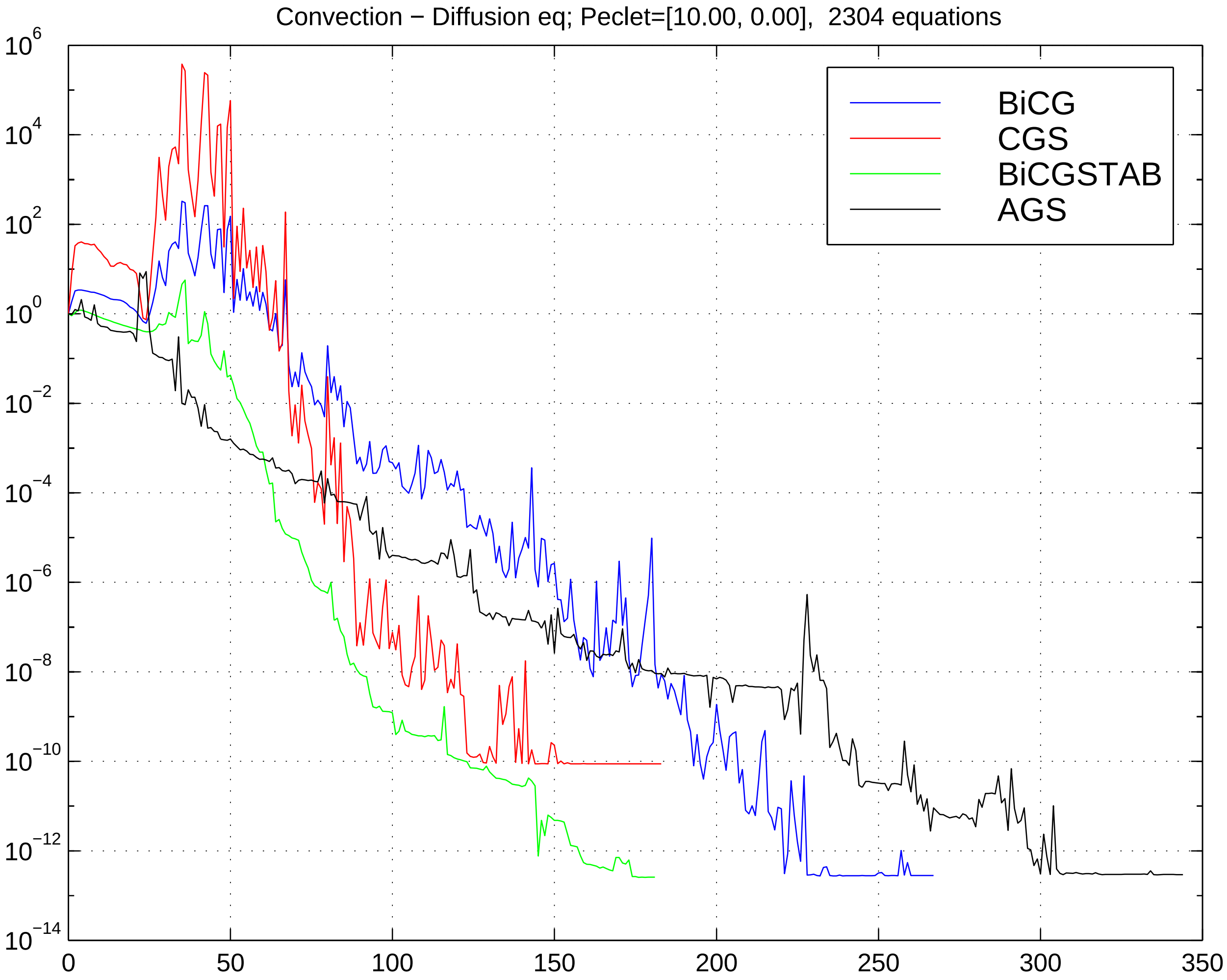
$$\widehat{\boldsymbol{x}}_{n+1} = \widehat{\boldsymbol{x}}_n + \widehat{\boldsymbol{v}}$$

BiCG versus CGS, simple Poisson





The price of squaring



Other way for calculation of ρ_n and σ_n

- Main property of φ_n: $\langle \vartheta, \varphi_n \rangle = 0$ for all ϑ of degree lower than n.
- A polynomial Ω of degree n satisfies $\Omega(t) = \gamma\varphi_n(t) + \vartheta(t)$, with ϑ of degree at most $n-1$.
- Alternative calculation:

$$\rho_n = \frac{\langle \Omega, \varphi_n \rangle}{\gamma}, \quad \sigma_n = \frac{\langle \Omega, t\varphi_n \rangle}{\kappa}$$

where $\Omega(t) = \gamma\varphi_n(t) + \vartheta(t) = \kappa\psi_n(t) + \upsilon(t)$.

- Only problem: the value of the scalefactors γ and κ.

Birth of BiCGSTAB 1

Old IDR algorithm had a polynomial connection:

$$\boldsymbol{r}_{2k} = \Omega_k(\boldsymbol{A})\varphi_k(\boldsymbol{A})\boldsymbol{r}_0, \quad \boldsymbol{r}_{2k+1} = \Omega_k(\boldsymbol{A})\psi_{k+1}(\boldsymbol{A})\boldsymbol{r}_0$$

where

$$\Omega_0(t) = 1,\ \Omega_k(t) = (1 - \omega_k t)\Omega_{k-1}(t),\ k = 1, 2, \ldots$$

(Here the polynomial ψ_n is not the BiCG-polynomial ψ_n.)

Polynomial Ω_n is build factor by factor. Each factor can be chosen the best contractor as possible *for the vector it operates on*.

Birth of BiCGSTAB 2

Alternative calculation:

$$\varphi_n(t) = (-1)^n \alpha_0 \alpha_1 \cdots \alpha_{n-1} t^n + \textit{lower degree}.$$

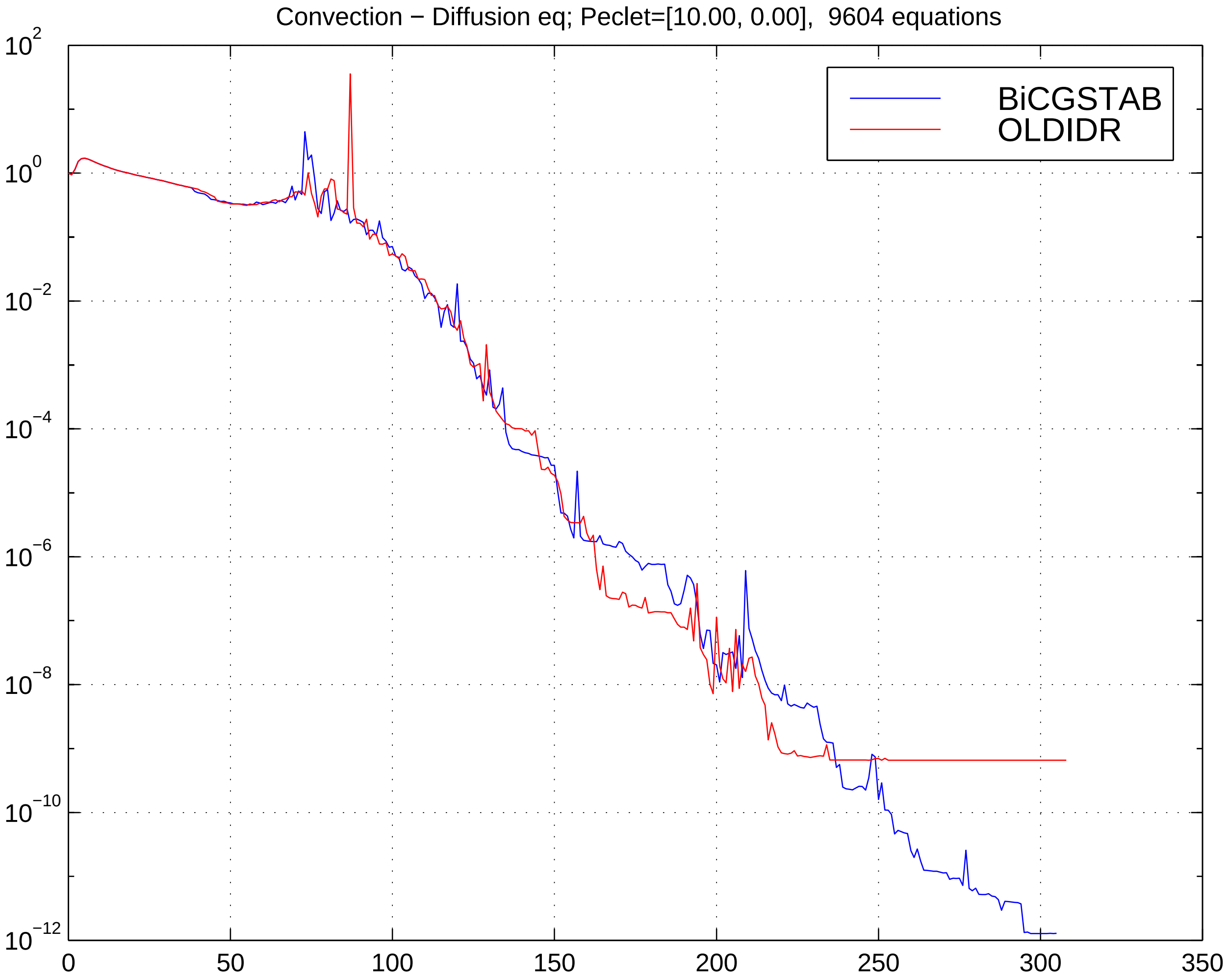
On the other hand

$$\Omega_n(t) = (-1)^n \omega_1 \omega_2 \cdots \omega_n t^n + \textit{lower degree}.$$

from which γ follows. A similar expresion holds for κ.

Now use $\Omega\varphi$ instead of φ^2 etc. The first algorithm that implements this was Bi-CGSTAB, in which ‘STAB’ means stabilization.

Old IDR versus BiCGSTAB



IDR and Bi-CGSTAB are different??

Observations:

- The methods behave identical, until IDR starts to suffer from instability. Natural: they produce the same residuals.
- A few peaks lead to loss of about 3 decimal digits compared to BiCGSTAB.
- For difficult problems, this occurred often,
- The author accepted the BiCGSTAB construction of the IDR-polynomials to be superior.

This, after all, was wrong. It was only a not so lucky implementation of the old IDR algorithm..

BiCGStab developments...

- Of course also BiCGSTAB sometimes suffered from problems.
- This led to the development of modifications and generalizations of BiCGSTAB.
- Very clever generalizations have been developed by Martin Gutknecht, and Gerard Sleijpen
- The author had other things on his mind, and went on with a not specifically Krylov-subspace-related life....

Zemke, and a short monologue

- 2006: Jens-Peter Zemke, from Hamburg, mails: What happened to IDR?
- Have to read carefully the 1980 version of the theorem, and the ancient history.
- Theorem used a space $\mathcal{S}$, not just $\boldsymbol{p}^{\perp}$.
- Serendipity moment: Why didn't I use more vectors $\boldsymbol{p}$, say s instead of 1???
- Because it costs $s+1$ matvecs per $\mathcal{G}_j$-space.
- But maybe there is more dimension reduction per $\mathcal{G}_j$
- .... Never thought about, must try... and call it IDR(s)

IDR-theorem again

Theorem 2 (IDR) *Let $\boldsymbol{A}$ be any matrix in $\mathbb{R}^{N\times N}$, let $\boldsymbol{v}_0$ be any nonzero vector in $\mathbb{R}^N$, let $\mathcal{G}_0$ be the full Krylov space $\mathcal{K}^N(\boldsymbol{A},\boldsymbol{v}_0)$, let $\mathcal{S}$ denote any (proper) subspace of $\mathbb{R}^N$, and let the sequence $\mathcal{G}_j$, $j = 1, 2, \ldots$ be defined by*

$$\mathcal{G}_j = (\boldsymbol{I} - \omega_j \boldsymbol{A})(\mathcal{G}_{j-1} \cap \mathcal{S})$$

where ω_j are nonzero numbers. Then

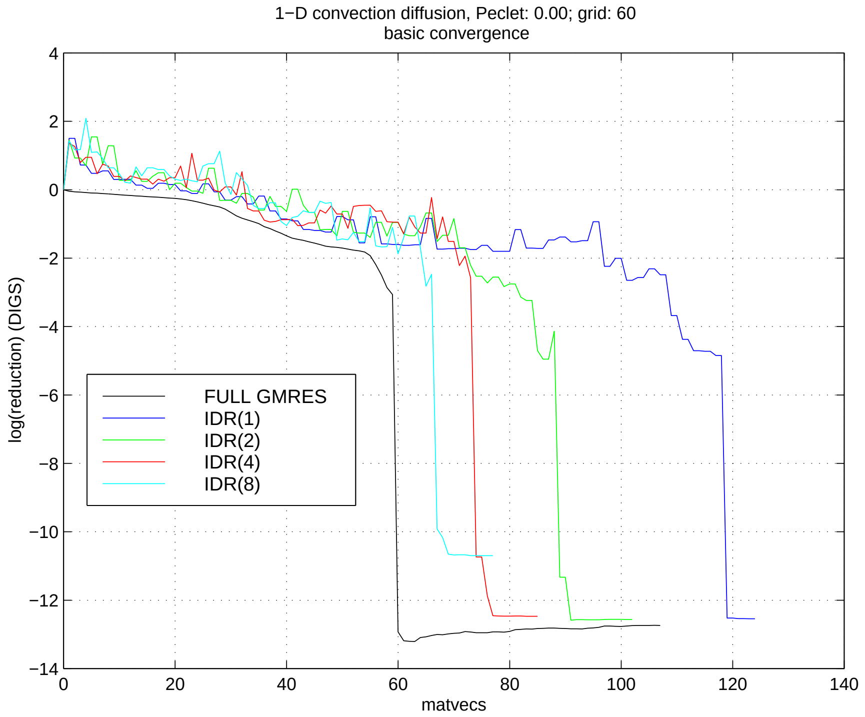
i: $\mathcal{G}_j \subset \mathcal{G}_{j-1}$, ii: $M \leq N$ exists such that

$\mathcal{G}_j = \mathcal{G}_M, \;\; j = M+1, M+2, \ldots$

Principle of IDR(s) algorithms

1. Suppose $\boldsymbol{P}$ is some $N \times s$ matrix $\boldsymbol{P}$,and let $\mathcal{S} = \mathcal{N}(\boldsymbol{P})$.
2. Suppose we have $s+1$ independent vectors $\boldsymbol{r}^{(n)}, \boldsymbol{r}^{(n-1)}, \boldsymbol{r}^{(n-2)}, \ldots, \boldsymbol{r}^{(n-s)}$ in $\mathcal{G}_{j-1}$.
3. Define $\boldsymbol{R}_n = [\boldsymbol{r}^{(n)}, \boldsymbol{r}^{(n-1)}, \boldsymbol{r}^{(n-2)}, \ldots, \boldsymbol{r}^{(n-s)}]$
4. Determine a solution of $\boldsymbol{P}^T \boldsymbol{R}_n \boldsymbol{c} = \boldsymbol{0}$, with $\sum c_j = 1$.
5. Then $\boldsymbol{R}_n \boldsymbol{c}$ is in $\boldsymbol{S} \cap \mathcal{G}_{j-1}$, and therefore
6. $(\boldsymbol{I} - \omega_j \boldsymbol{A})(\boldsymbol{R}_n \boldsymbol{c})$ is in G_j.
7. Since $\mathcal{G}_j \subset \mathcal{G}_{j-1}$, this can be repeated to generate more vectors in $\mathcal{G}_j$.
8. Since $\sum c_j = 1$, an x-update can be made.

Termination of IDR(s) on 60×60 system.

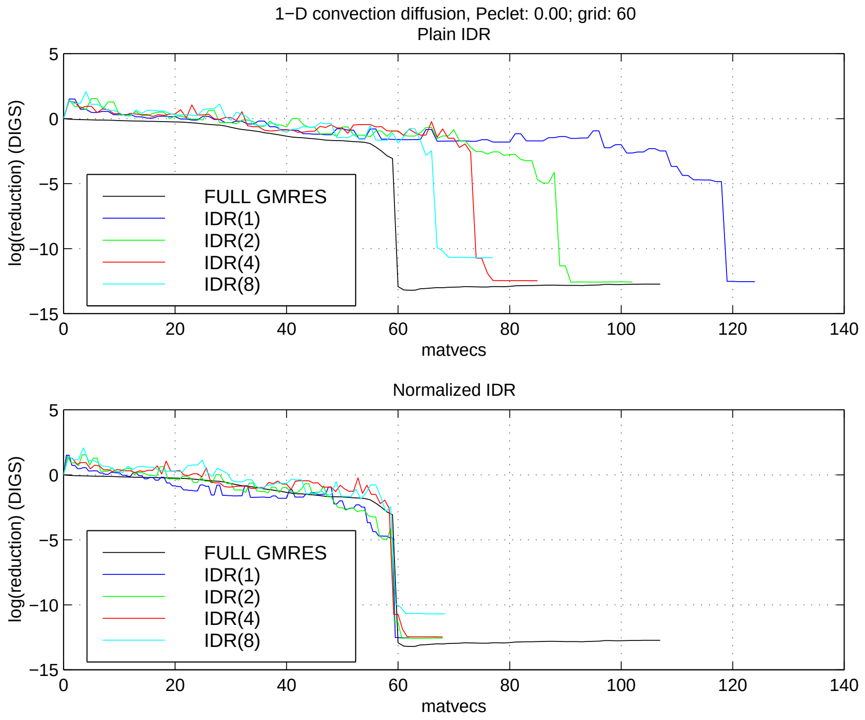


Apparently we have a dimension reduction of s at every $s+1$ steps.

Check that out, by rescaling the matvec-axis.

Scaling the ‘matvecs-axis’ (1)

Finite behavior for IDR(s): $\frac{s+1}{s}N$ steps. This suggests a rescaling of the matvecs count with a factor $\frac{s}{s+1}$:



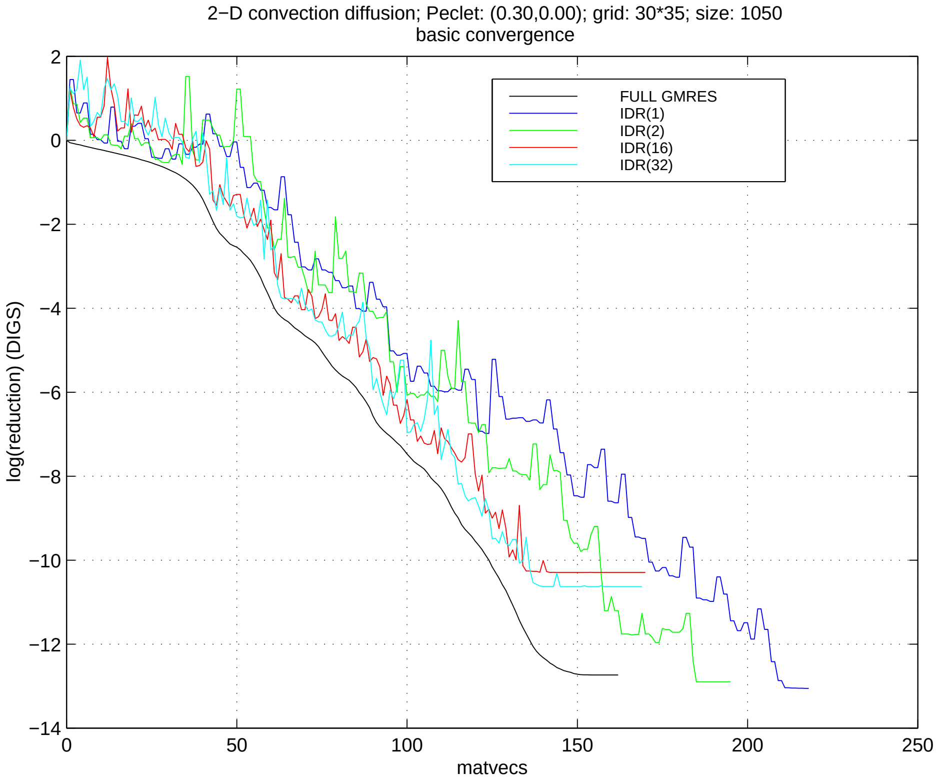
Upper picture is unscaled

Lower picture:

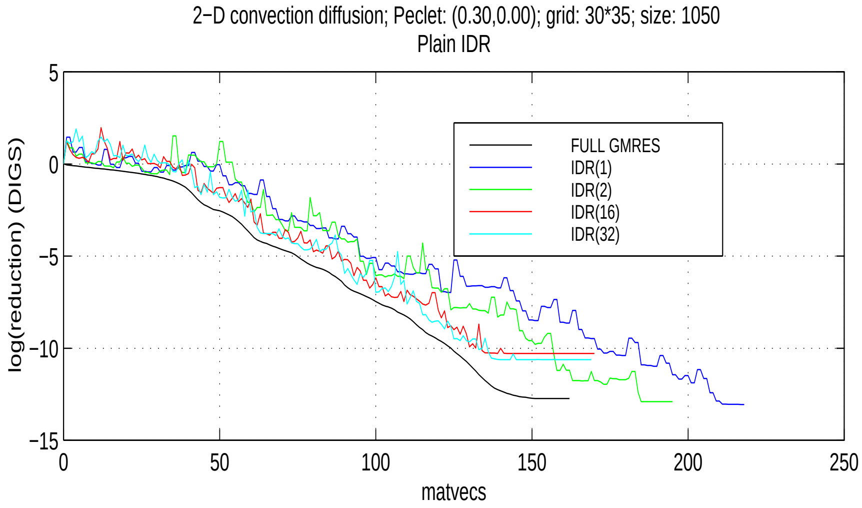
Horizontal axis displays $s/(s+1)\times$ #matvecs

Serendipity moment: Do the same rescaling on convergence plots

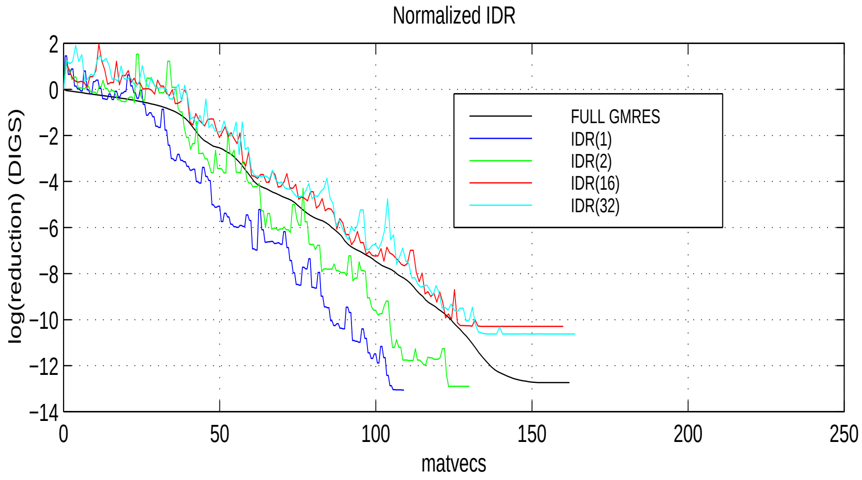
Convergence 2-D problem’



Scaling the matvecs-axis



2−D convection diffusion; Peclet: (0.30,0.00); grid: 30*35; size: 1050
Plain IDR



Normalized IDR

Upper picture: unscaled.

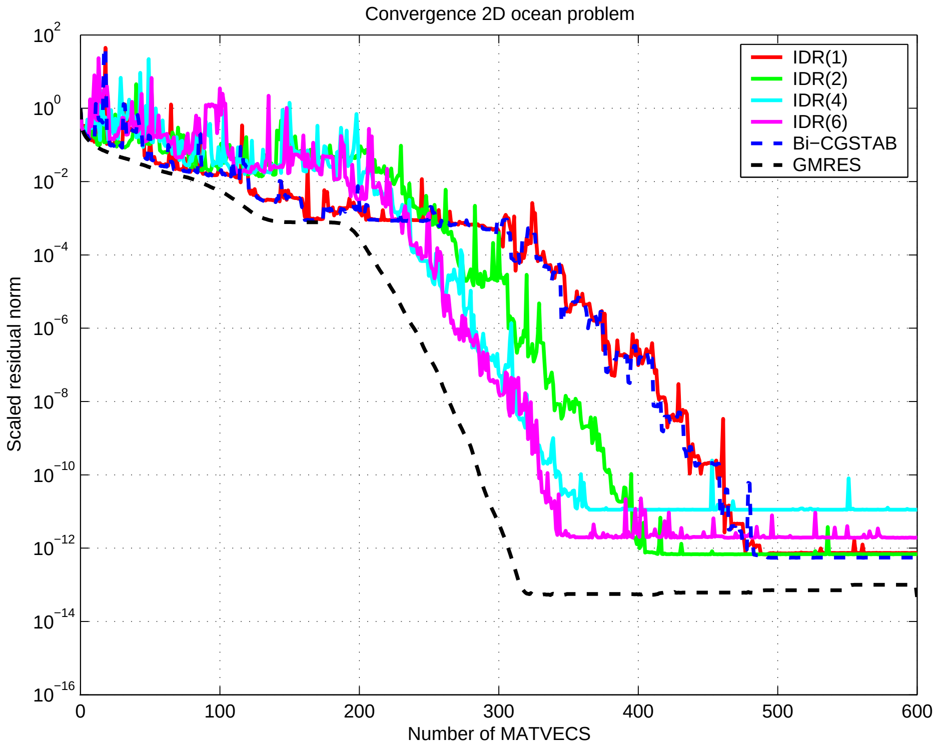
Lower picture: For higher s, the convergence behaviour is similar to the finite behaviour.

Do not be mislead: $s = 1$ and $s = 2$ show only half and two third of the work respectively.

IDR(s) on a realistic problem

Problem: Convection diffusion equation from oceanography

Size: 42248 equations. **Sparseness:** About 300000 nonzeros.



The convergence at different s is similar as in the finite termination behaviour.

What have we seen?

In the picture with scaled matvec-axis, IDR(1) and IDR(2) show better behaviour than the rest. But appearances are deceptive, the actual work is 2 resp $\frac{3}{2}$ times as much.

IDR(s) for higher s-values appears to be comparable to GMRES which is the best (with respect to matvec-economy)

However: GMRES has $O(n)$ inprods/vector-updates at step n, leading to $O(n^2)$ total extra work.

Begin of exlanation

Polynomials again! If $n \in [l(s+1), l(s+1)+s]$, the residuals satisfy

$$\boldsymbol{r}^{(n)} = \Omega_l(\boldsymbol{A})\Psi_{n-l}(\boldsymbol{A})\boldsymbol{r}^{(0)}$$

where

$$\Omega_l(t) = \prod_{k=1}^{l}(1-\omega_k t), \quad \Psi_{n-l}(t) = 1 + \sum_{j=1}^{n-l} c_j t^j$$

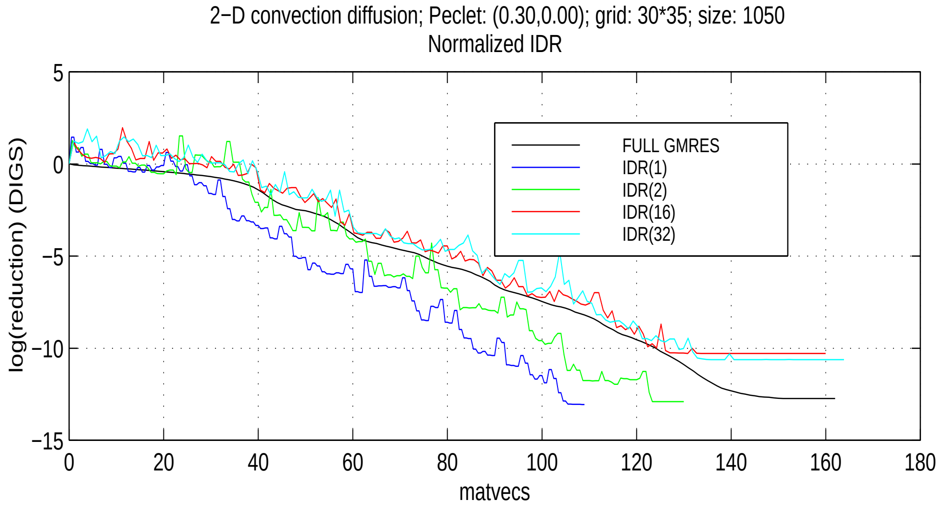
similar as in the old IDR algorithm. The factors $(\boldsymbol{I} - \omega_l \boldsymbol{A})$ in $\Omega_l(\boldsymbol{A})$ are constructed as damping factor, for stabilization (just as in old IDR).

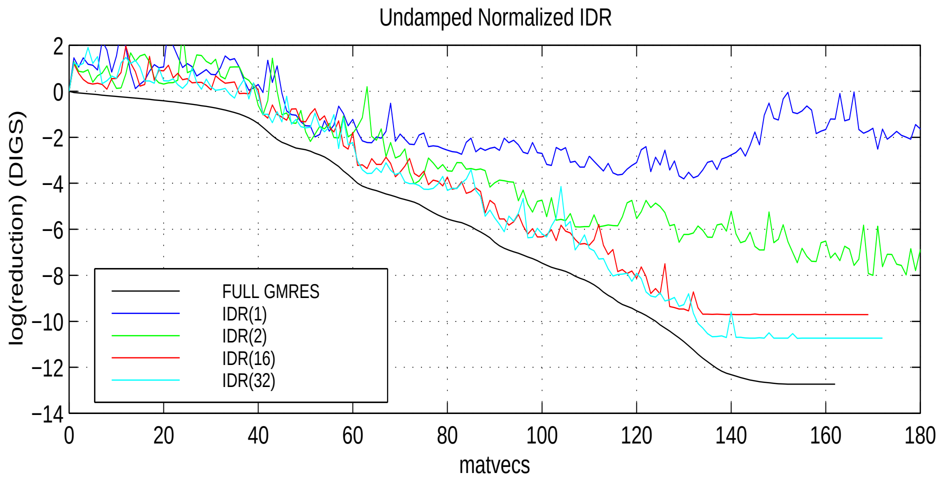
Continuation of explanation

In IDR(s), roughly speaking in every $s + 1$ steps, one damping factor is applied. Therefore low values of s will show more damping.
To explain the rest of the behavior, we need an interpretation of the other polynomials Ψ_{n-l}. How?
By removing the Ω-factors. This is not a trivial job, the process is not very stable, and rather inefficient.

Have pity with the programmer.

Removing the Ω-factors





Upper plot: Only scaled matvecs.

Lower plot: $\boldsymbol{r}_l = \Psi_n(\boldsymbol{A})\boldsymbol{r}_0$. So the Ω-factors are removed as well.

These plots are generalizations of BiCG, except for high-numbered iteration steps, due to the instable implementation

Galerkin interpretation (1)

- If $n = l(s+1)$, the polynomials Ψ_{n-l} satisfy

$$\boldsymbol{p}_r^H \boldsymbol{A}^j \Psi_{n-l}(\boldsymbol{A})\boldsymbol{r}^{(0)} = 0, \;\; j = 0, 1, \ldots, l-1, \;\; r = 1, 2, \ldots, s$$

$$\Psi(0) = 1$$

- This can be interpreted as a Galerkin approximation for an overdetermined system

$$\boldsymbol{r}^{(0)} + \sum_{j=1}^{n-l} c_j \boldsymbol{A}^j \boldsymbol{r}^{(0)} = \boldsymbol{0}$$

- with *testvectors* $(\boldsymbol{A}^T)^j \boldsymbol{p}_r$, $r = 1, 2, \ldots, s, \;\; j = 0, 1, \ldots, l-1$
- We call this Krylov Galerkin.

Galerkin interpretation (2)

The vectors $\boldsymbol{A}^j\boldsymbol{r}_0$ act as columns in a modelmatrix $\boldsymbol{M}$.
If we choose these as testvectors as well: We get the least squares solution:

$$\|\boldsymbol{r}^0 + \sum_{j=1}^{n-l} c_j \boldsymbol{A}^j \boldsymbol{r}^{(0)}\| \quad \text{is minimal}$$

GMRES is an excellent practical implementation of this principle,

with only one disadvantage: a full depth recursion, with quadratically growing overhead.

On Galerkin methods (1)

- Classical overdetermined linear system:

$$\boldsymbol{Mc} = \boldsymbol{b}, \quad \text{where } \boldsymbol{M} \text{ is an } N \times k \text{ matrix, } N > k$$

- Galerkin approximation: Solve $\boldsymbol{T}^H\boldsymbol{Mc} = \boldsymbol{T}^H\boldsymbol{b}$ instead, for some $N \times k$ *testmatrix* $\boldsymbol{T}$.
  Formal solution: $\boldsymbol{c} = (\boldsymbol{T}^H\boldsymbol{M})^{-1}\boldsymbol{T}^H\boldsymbol{b}$.
- Residual: $\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{Mc} = (\boldsymbol{I} - \boldsymbol{P})\boldsymbol{b}$, with $\boldsymbol{P} = \boldsymbol{M}(\boldsymbol{T}^H\boldsymbol{M})^{-1}\boldsymbol{T}^H$
  .
- $\boldsymbol{P}$ is an oblique projection, satisfying $\boldsymbol{P}^2 = \boldsymbol{P}$.

On Galerkin methods (2)

- If the test space coincides with the model space, i.e. $\mathcal{R}(\boldsymbol{T}) = \mathcal{R}(\boldsymbol{M})$, we get the least squares solution.

$$\widehat{\boldsymbol{r}} = (\boldsymbol{I} - \widehat{\boldsymbol{P}})\boldsymbol{b}, \quad \text{with } \widehat{\boldsymbol{P}} = \boldsymbol{M}(\boldsymbol{M}^H\boldsymbol{M})^{-1}\boldsymbol{M}^H$$

- The residual $\widehat{\boldsymbol{r}}$ is perpendicular to the columns of $\boldsymbol{M}$, since $\widehat{\boldsymbol{P}} = \widehat{\boldsymbol{P}}^H$.
- $\|\widehat{\boldsymbol{r}}\|$ is not larger than $\|\boldsymbol{r}\|$ for any residual obtained by any other Galerkin procedure.
- How much smaller???

A funny property of projections

All projection operators for Galerkin methods have the same column space, $\mathcal{R}(M)$.

Theorem: If $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ are projections such that $\mathcal{R}(\boldsymbol{P}_1) = \mathcal{R}(\boldsymbol{P}_2)$, then $\boldsymbol{P}_1\boldsymbol{P}_2 = \boldsymbol{P}_2$, and $(\boldsymbol{I} - \boldsymbol{P}_1)(\boldsymbol{I} - \boldsymbol{P}_2) = \boldsymbol{I} - \boldsymbol{P}_1$.

Proof:

Let $\boldsymbol{x}$ be arbitrary. Then

$\boldsymbol{P}_2\boldsymbol{x} \in \mathcal{R}(\boldsymbol{P}_2) \Longrightarrow \boldsymbol{P}_2\boldsymbol{x} \in \mathcal{R}(\boldsymbol{P}_1) \Longrightarrow \boldsymbol{P}_2\boldsymbol{x} = \boldsymbol{P}_1\boldsymbol{y}$ for some $\boldsymbol{y}$

Then $\boldsymbol{P}_1\boldsymbol{P}_2\boldsymbol{x} = \boldsymbol{P}_1\boldsymbol{P}_1\boldsymbol{y} = \boldsymbol{P}_1\boldsymbol{y} = \boldsymbol{P}_2\boldsymbol{x}$ and finally

$$(\boldsymbol{I} - \boldsymbol{P}_1)(\boldsymbol{I} - \boldsymbol{P}_2) = \boldsymbol{I} - \boldsymbol{P}_1 - \boldsymbol{P}_2 + \boldsymbol{P}_1\boldsymbol{P}_2 = \boldsymbol{I} - \boldsymbol{P}_1$$

Application of funny property

Let $\boldsymbol{P}$ denote some 'Galerkin projector' on $\mathcal{R}(\boldsymbol{M})$, and let $\widehat{\boldsymbol{P}}$ denote the least squares projector on the same space.

The Galerkin residual satisfies $\boldsymbol{r} = (\boldsymbol{I} - \boldsymbol{P})\boldsymbol{b}$, the least squares residual satisfies $\widehat{\boldsymbol{r}} = (\boldsymbol{I} - \widehat{\boldsymbol{P}})\boldsymbol{b}$.

According to the funny property we have

$$\boldsymbol{r} = (\boldsymbol{I} - \boldsymbol{P})\boldsymbol{b} = (\boldsymbol{I} - \boldsymbol{P})(\boldsymbol{I} - \widehat{\boldsymbol{P}})\boldsymbol{b} = (\boldsymbol{I} - \boldsymbol{P})\widehat{\boldsymbol{r}}$$

The surplus $d\boldsymbol{r}$ to the least-squares residual then satisfies:

$$d\boldsymbol{r} = \boldsymbol{r} - \widehat{\boldsymbol{r}} = -\boldsymbol{P}\widehat{\boldsymbol{r}} \perp \widehat{\boldsymbol{r}}$$

An estimate for the Galerkin residual

The least squares residual is perpendicular to $\mathcal{R}(\widehat{\boldsymbol{P}})$ and thus to $\mathcal{R}(\boldsymbol{P})$. Therefore $d\boldsymbol{r} \perp \widehat{\boldsymbol{r}}$.
Let the k columns of $\boldsymbol{Q}_1$ and the $N-k$ columns of $\boldsymbol{Q}_2$ be orthonormal bases for $\mathcal{R}(\boldsymbol{P})$ and $\mathcal{R}(\boldsymbol{P})^\perp$ respectively. So $\boldsymbol{Q}_1^H\boldsymbol{Q}_2 = \boldsymbol{O}$, $\boldsymbol{Q}_1^H\boldsymbol{Q}_1 = \boldsymbol{I}^k$, and $\boldsymbol{Q}_2^H\boldsymbol{Q}_2 = \boldsymbol{I}^{N-k}$. Then

- 1) $\boldsymbol{P} = \boldsymbol{Q}_1(\boldsymbol{T}^H\boldsymbol{Q}_1)^{-1}\boldsymbol{T}^H$
- 2) $\widehat{\boldsymbol{r}} = \boldsymbol{Q}_2\boldsymbol{z}$ for some $\boldsymbol{z}$, with $\|\boldsymbol{z}\| = \|\widehat{\boldsymbol{r}}\|$
- 3) The residual surplus can be written as

$$d\boldsymbol{r} = \boldsymbol{P}\widehat{\boldsymbol{r}} = \boldsymbol{Q}_1(\boldsymbol{T}^H\boldsymbol{Q}_1)^{-1}(\boldsymbol{T}^H\boldsymbol{Q}_2)\boldsymbol{z}$$

where $\boldsymbol{z}$ can be any vector in $\mathbb{R}^{N-k}$

Back to Krylov Galerkin

In general not much can be said about $\|\boldsymbol{Q}_1(\boldsymbol{T}^H\boldsymbol{Q}_1)^{-1}\boldsymbol{T}^H\boldsymbol{Q}_2\boldsymbol{z}\|$. A clever choice seems to be $\boldsymbol{T} = \boldsymbol{M}$, because in that case $\boldsymbol{T}^H\boldsymbol{Q}_2 = \boldsymbol{O}$. But this is GMRES, and that will be expensive in overhead.

In the IDR(s) algorithms, we usually choose random vectors $\boldsymbol{p}_k$, since we do not (yet) know better.

From the pictures can be seen that for large s, say $s \geq 16$, there is not much difference in convergence behaviour of the Krylov Galerkin method with random $\boldsymbol{p}_k$ vectors.

So we'll consider the case $s = \infty$, meaning testvectors that are completely randomly chosen.

Random testvectors (1)

- Let $\boldsymbol{T}$ be an $N \times k$ matrix, of which all entries are stochastically independent, and normally distributed $N(0,1)$. We're interested in $\|d\boldsymbol{r}\| = \|\boldsymbol{Q}_1(\boldsymbol{T}^H\boldsymbol{Q}_1)^{-1}\boldsymbol{T}^H\boldsymbol{Q}_2\boldsymbol{z}\|$
- $\boldsymbol{Q}_1$ and $\boldsymbol{Q}_2$ are complementary, i.e. the compound matrix $\boldsymbol{Q} = [\boldsymbol{Q}_1 \,|\, \boldsymbol{Q}_2]$ square and unitary. Then $\widetilde{\boldsymbol{T}} = \boldsymbol{Q}^H\boldsymbol{T}$ has the same joint probability distribution as $\boldsymbol{T}$.
- Let $\widetilde{\boldsymbol{T}}_j^H = \boldsymbol{T}\boldsymbol{Q}_j$, for $j = 1, 2$, then all entries of these matrices are stochastically independent, and $N(0,1)$ distributed.
- We may write $\|d\boldsymbol{r}\| = \|\boldsymbol{Q}_1\widetilde{\boldsymbol{T}}_1^H\widetilde{\boldsymbol{T}}_2\boldsymbol{z}\| = \|\widetilde{\boldsymbol{T}}_1^H\widetilde{\boldsymbol{T}}_2\boldsymbol{z}\|$, with $\boldsymbol{z} \in \mathbb{R}^{N-k}$ a given vector that is completely determined by the least squares solution.

Random testvectors (2)

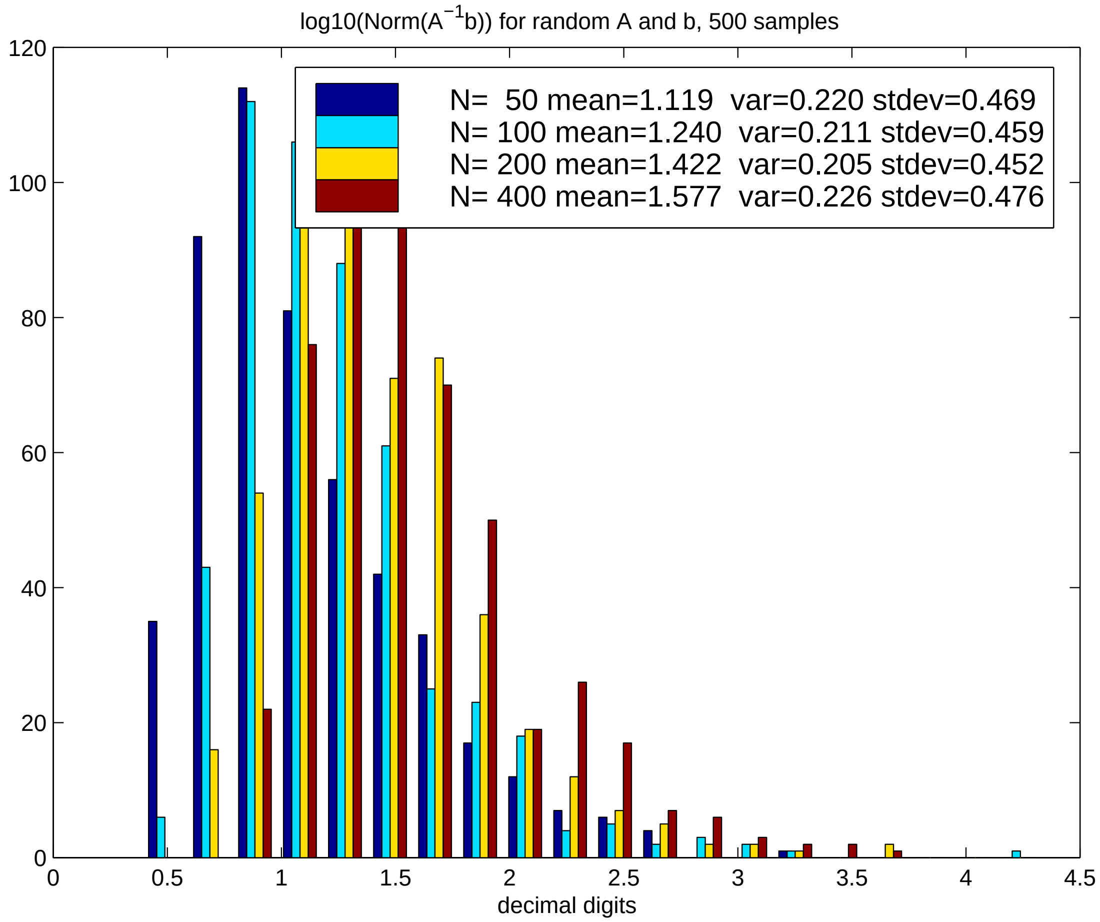
Denote matrices and vectors of which all entries are stochastically independent, and distributed normally $N(0,1)$ by Gaussian matrices and vectors respectively.

- Let $\boldsymbol{u} = \widetilde{\boldsymbol{T}}_2\boldsymbol{z}$. According to rather elementary probability theory, the entries of $\boldsymbol{u}$ are stochastically independent normally distributed $N(0, \|\boldsymbol{z}\|)$. Or we can say: $\boldsymbol{u} = \|\boldsymbol{z}\|\boldsymbol{v}$, where $\boldsymbol{v}$ is a Gaussian vector in $\mathbb{R}^k$.
- Then $d\boldsymbol{r} = \|\widehat{\boldsymbol{r}}\|.\widetilde{\boldsymbol{T}}_1^H\boldsymbol{v}$,
- where $\widetilde{\boldsymbol{T}}_1^H$ and $\boldsymbol{v}$ are a Gaussian $k \times k$ matrix, and a Gaussian vector in $\mathbb{R}^k$ respectively.

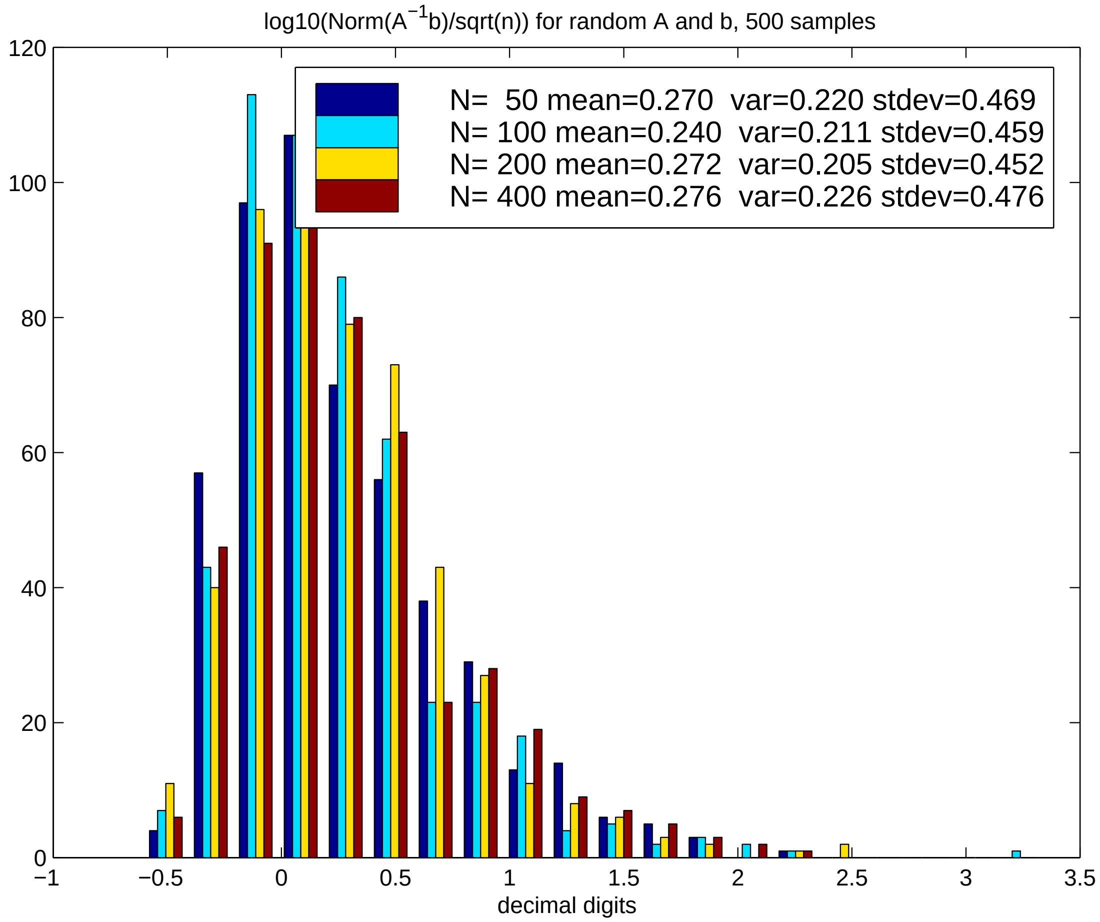
Random testvecors (3)

- For large k the inverse norm of the Gaussian matrix $\|\widetilde{\boldsymbol{T}}_1^{-1}\|$ is of order $O(\sqrt{k})$ (Alan Edelman). The norm of a Gaussian vector in $\mathbb{R}^k$ also is $O(\sqrt{k})$.
- It follows $\|d\boldsymbol{r}\| \leq \|\widehat{\boldsymbol{r}}\|.\|(\widetilde{\boldsymbol{T}}_1^H)^{-1}\|.\|\boldsymbol{v}\| \leq C\|\widehat{\boldsymbol{r}}\|k$ for moderate value of C.
- Experiments showed something like $\|d\boldsymbol{r}\| = O(\sqrt{k})$ instead.
- Therefore a statistical experiment was done on the distribution of $\|\boldsymbol{B}^{-1}\boldsymbol{v}\|$, for Gaussian $\boldsymbol{B}$ and $\boldsymbol{v}$.

Experimental Distribution



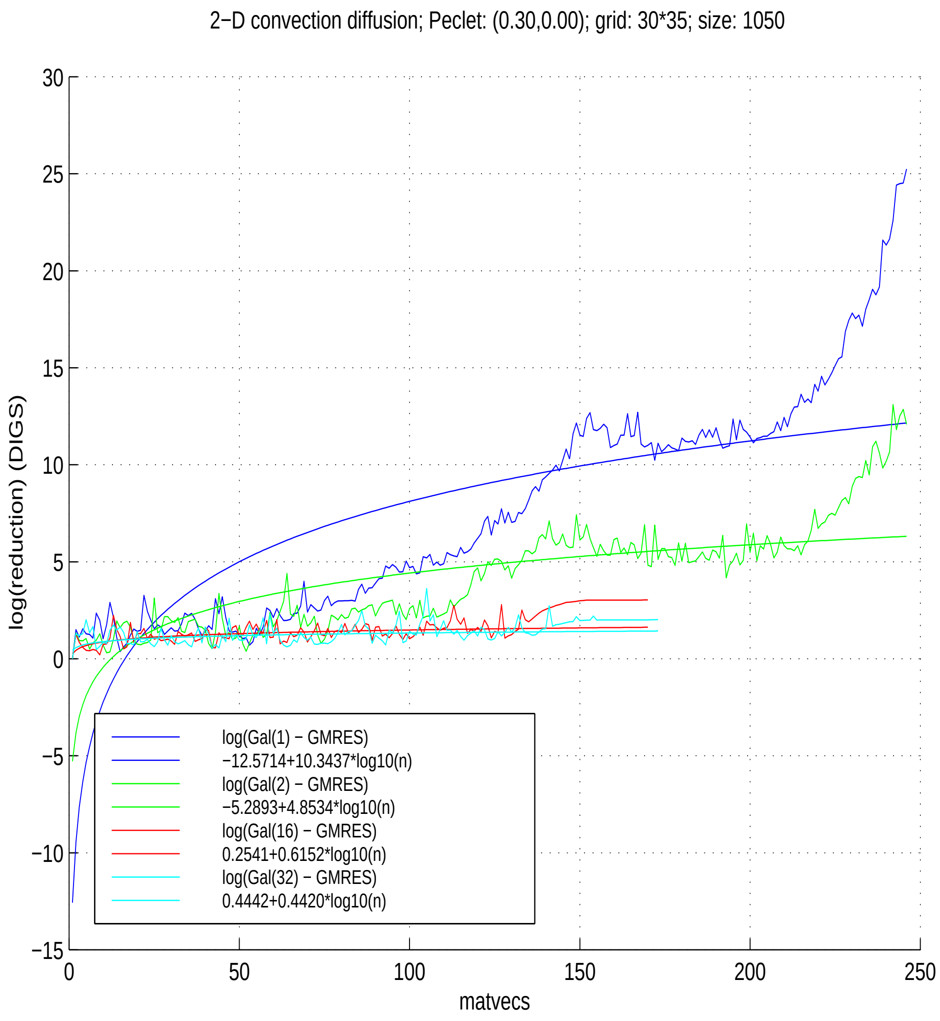
Scaled experimental Distribution



What does this all mean?

- The 10-logarithm of $d\boldsymbol{r}$ is expected to behave like $\log_{10}(\|\widehat{\boldsymbol{r}}\|) + 0.27 + \log_{10}(k)/2$.
- With probability close to one we have $\log_{10}(\|d\boldsymbol{r}\|) \leq \log_{10}(\|\widehat{\boldsymbol{r}}\|) + 0.27 + \log_{10}(k)/2 + 1.4$
- So random Galerkin is expected to run about $0.27 + \log_{10}(k)/2$ decimal digits behind least squares at step k. At most 1.5 decimal digits extra may be lost.
- For IDR(s), with $s \geq 16$, a same kind of result may be expected. As we have seen a few sheets before.

Result of experiment.



Reduced IDR(s)-GMRES, logarithmic graph.

The reduced residual-logarithms are fitted with $C_0 + C_1 \log_{10}(k)$.

For $s = 1$ and $s = 2$, we are far from the 'full random' property. Hence a bad fit.

s	C_0	C_1
16	0.2541	0.6152
32	0.4442	0.4420

Final remarks

The author thinks that serendipity is an important part of scientific research, and at least it is an extremely satisfying part. According to Peter Wynn, ‘numerical analysis is much of an experimental science’, and in the IDR-CGS-IDR(s) development, the experimental part was the main source of serendipity. So the numerical mathematician should never hesitate to do numerical experiments, nor hesitate to look not only to his/her results, but also the non-results. There may be something in it!

Without Martin van Gijzen, this story wouldn’t have been told before I was 80 years old. Thank you Martin.