MA 580; Iterative Methods for Linear Equations

C. T. Kelley
NC State University
tim_kelley@ncsu.edu
Version of October 10, 2016
Read Chapters 2 and 3 of the Red book.

NCSU, Fall 2016
Part VIb: Krylov Methods for Linear Equations: GMRES
Don’t forget the notation

- We’re solving $Ax = b$ for $x^* = A^{-1}b$ in $R^N$.
- $\{x_n\}_{n=0}^\infty$ is the sequence of iterates.
- $x_0$ is the initial iterate (not guess).
- $e = x^* - x$ is the error.
- $r = b - Ax = Ae$ is the residual.
What Krylov Methods Do

- Krylov iterative methods obtain \( \mathbf{x}_n \) from the history of the iteration.
- The ones with theory do this by minimizing an error or residual function over the affine space

\[
\mathbf{x}_0 + \mathcal{K}_k
\]

- \( \mathbf{x}_0 \) is the initial iterate
- \( \mathcal{K}_k \) is the \( k \)th Krylov subspace

\[
\mathcal{K}_k = \text{span}(\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \ldots, \mathbf{A}^{k-1}\mathbf{r}_0)
\]

for \( k \geq 1 \).
GMRES and Conjugate Gradient (CG)

These two methods can be expressed in terms of \textit{minimization principles}

In GMRES (Generalized Minimum Residual), the $k$th iteration $\mathbf{x}_k$ minimizes the residual over $\mathbf{x}_0 + \mathcal{K}_k$

$$\| \mathbf{b} - \mathbf{A}\mathbf{x}_k \| = \min_{\mathbf{x} \in \mathbf{x}_0 + \mathcal{K}_k} \| \mathbf{b} - \mathbf{A}\mathbf{x} \|$$

for $\| \cdot \| = \| \cdot \|_2$. For CG, $\mathbf{A}$ must be spd and $\mathbf{x}_k$ minimizes the $\mathbf{A}$-norm of the error

$$\| \mathbf{x}^* - \mathbf{x} \|_\mathbf{A} = \min_{\mathbf{x} \in \mathbf{x}_0 + \mathcal{K}_k} \| \mathbf{x}^* - \mathbf{x} \|_\mathbf{A}$$

where

$$\| \mathbf{v} \|_\mathbf{A}^2 = \mathbf{v}^T \mathbf{A}\mathbf{v}.$$
Analysis of GMRES

If \( x \in x_0 + K_k \) then

\[
    r = b - Ax = b - Ax_0 - \sum_{j=1}^{k} \gamma_j A^j r_0 \equiv p(A)r_0
\]

where \( p \in P_k \), the set of \( k \) degree residual polynomials.

\[
    P_k = \{ p \mid p \text{ is a polynomial of degree } k \text{ and } p(0) = 1. \}
\]

This simple observation is the key to analysis of Krylov methods.
Theorem: Let $A$ be nonsingular and let $x_k$ be the $k$th GMRES iteration. Then for all $\bar{p} \in \mathcal{P}_k$

$$\|r_k\| = \min_{p \in \mathcal{P}_k} \|p(A)r_0\| \leq \|\bar{p}(A)r_0\|.$$
Proof of Theorem

Let $x_k$ the the $k$th GMRES iteration. Then there is $p_k \in P_k$ such that

$$r_k = b - Ax_k = p_k(A)r_0$$

Since any $x \in x_0 + K_k$ satisfies

$$r = b - Ax = \bar{p}(A)r_0$$

for some $\bar{p} \in P_k$, the minimization principle implies that

$$\|r_k\|_2 = \min_{p \in P_k} \|p(A)r_0\| = \|\bar{p}_k(A)r_0\| \leq \|b - Ax\| = \|\bar{p}(A)r_0\|.$$
How to Use This Theorem

- Connect properties of the matrix to a polynomial you understand.
- Manufacture a residual polynomial $\tilde{p}$ from that.
- Get an upper bound from

$$\|r_k\| \leq \|\tilde{p}(A)r_0\| \leq \|\tilde{p}(A)\|\|r_0\|$$
Corollary: Let $A$ be nonsingular. Then the GMRES algorithm will find the solution within $N$ iterations.

Proof: The characteristic polynomial of $A$ is $p(z) = \text{det}(A - zI)$. $p$ has degree $N$, $p(0) = \text{det}(A) \neq 0$ since $A$ is nonsingular, and so

$$\bar{p}_N(z) = p(z)/p(0) \in \mathcal{P}_N$$

is a residual polynomial. The Cayley-Hamilton theorem says that $\bar{p}_N(A) = 0$, and so

$$\|r_N\| \leq \|\bar{p}_N(A)\|\|r_0\| = 0.$$
Corollary: If \( \| I - A \| \leq \rho < 1 \) then

\[
\| r_k \| \leq \rho^k \| r_0 \|_2.
\]

Proof: Let \( \bar{p}_k = (1 - z)^k \) and use the theorem.
Corollary: If \( \mathbf{b} \) is a linear combination of \( k \) eigenvectors of \( \mathbf{A} \), and let \( \mathbf{x}_0 = 0 \), then GMRES will converge it at most \( k \) iterations.

Proof: Let \( \mathbf{b} = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \) where \( \mathbf{A} \mathbf{u}_i = \lambda_i \mathbf{u}_i \). Let

\[
\bar{p} = \prod (\lambda_i - z)/\lambda_i.
\]

Then \( \bar{p}(\mathbf{A})\mathbf{r}_0 = \bar{p}(\mathbf{A})\mathbf{b} = 0 \) (can you see why?)

We’re done since

\[
\|\mathbf{r}_k\| \leq \|\bar{p}(\mathbf{A})\mathbf{r}_0\| = 0.
\]
A is diagonalizable if there is a nonsingular (possibly complex!) matrix $V$ such that

$$A = V \Lambda V^{-1}.$$ 

If $A$ is diagonalizable and $p$ is a polynomial then

$$p(A) = \sum_{j=0}^{m} a_j \gamma_j A^j = \sum_{j=0}^{m} a_j (V \Lambda V^{-1})^j = V \sum_{j=0}^{m} a_j \Lambda^j V^{-1} = V p(\Lambda) V^{-1}$$

So

$$\|p(A)\| \leq \|V\| \|p(\Lambda)\| \|V^{-1}\| = \kappa(V) \max_{\lambda \in \sigma(A)} |p(\lambda)|$$
We just proved . . .

Theorem: Let \( A = V \Lambda V^{-1} \) be a nonsingular diagonalizable matrix. Let \( x_k \) be the \( k \)th GMRES iterate. Then for all \( \bar{p}_k \in \mathcal{P}_k \)

\[
\frac{\|r_k\|_2}{\|r_0\|_2} \leq \kappa_2(V) \max_{z \in \sigma(A)} |\bar{p}_k(z)|.
\]
Easy Results for Diagonalizable $A$: 1

If $A$ has $m$ distinct eigenvalues then GMRES will terminate in at most $m$ iterations.

Proof: Use

$$p(z) = \prod_{i=1}^{m} \left( \frac{\lambda_i - \lambda}{\lambda_i} \right)$$

$p(0) = 1$ so $p \in P_k$. Since $p(\lambda_i) = 0$ for all $i$, $r_N = 0$. This proof is (1) very easy and (2) typical of the way one thinks about Krylov methods.
Easy Results for Diagonalizable $A$: II

Let $x_0 = 0$ (so $r_0 = b$) and assume that

- $\sigma(A) \subset (9, 11)$
- $\kappa(V) = 100$

Then if we let $\bar{p}_k(z) = (10 - z)^k/10^k$ we see that

$$\frac{\|r_k\|_2}{\|b\|_2} \leq \kappa(V)\|p_k(\Lambda)\| \leq (100)10^{-k} = 10^{2-k}.$$ 

So $\|r_k\| \leq \eta\|b\|$ when

$$k > 2 - \log_{10}(\eta).$$

This tells us that an approximate inverse preconditioner could be useful.
For this example, how many iterations do you need to reduce the residual by a factor of $10^6$.
Use the formula with $\eta = 10^{-6}$ and get

$$k > 2 - \log_{10}(\eta) = 8.$$
More clustering stuff for diagonalizable $A$

Suppose

- $\kappa(V) = 1$ (eg $V$ is orthogonal)
- $\sigma(A) \subset (9, 11) \cup (19, 21) \cup \{7\}$

We need a residual polynomial that takes care of both clusters and the isolated point 7. How about

$$\bar{p} = \frac{7 - z}{7} \left( \frac{10 - z}{10} \frac{20 - z}{20} \right)^k \in \mathcal{P}_{2k+1}.$$ 

Clearly $\bar{p}(7) = 0$. 
What about the clusters?

If $z \in (9,11)$, then

$$|\bar{p}(z)| \leq \frac{11 - 7}{7} \frac{1}{10^k} \frac{11^k}{20^k} = \frac{11^k}{200^k} \leq 1/9^k.$$  

If $z \in (19,21)$, then

$$|\bar{p}(z)| \leq \frac{21 - 7}{7} \frac{11^k}{10^k} \frac{1}{20^k} = 2 \frac{11^k}{200^k} \leq 2/9^k.$$  

Bottom line: $\bar{p} \in \mathcal{P}_{2k+1}$ and

$$\max_{z \in \sigma(A)} |\bar{p}(z)| \leq 2/9^k$$
For this example, how many iterations do you need to reduce the residual by a factor of $10^6$.

Answer: $2k + 1$ where $2/9^k < 10^{-6}$.

- So $-k \log_{10} 9 + \log_{10} 2 < -6$
- and so $k > (6 + \log_{10} 2)/\log_{10} 9 \approx 6.6$

So $k = 7$ works and the answer is $2k + 1 = 15$. 
Normal Matrices

- **A** is normal if $A^T A = AA^T$.
- **A** normal implies $A = V \Lambda V^\#$.
  - Here $V$ (complex!) is unitary (i.e. $V^{-1} = V^\#$)
  - $V^\#$ is the complex conjugate transpose.
- **A** normal implies $\kappa(V) = 1$
General Observations for diagonalizable matrices

- The heuristic is that if the eigenvalues are grouped into a few clusters the iteration will perform well.
- If $\kappa(V)$ is large, then $\sigma(A)$ does not tell the whole story.
- If the eigenvalues are clustered near 1, then GMRES is very happy and $A$ is well-conditioned.

If $A$ is not diagonalizable then clusters don’t help you.
Preconditioning means to replace $Ax = b$ with

$$BAx = Bb \text{ (left)}$$

or

$$ABy = b \text{ (right)}, \text{ and then } x = By$$

and solve the preconditioned equation with GMRES. The preconditioner $B$ should be

- very inexpensive matrix-vector products
- be a good approximate inverse of (part) of $A$

Examples coming later.
Left Preconditioning

Solve

$$BAx = Bb$$

so

- solution to preconditioned equation is still $x$
- preconditioned residual $Bb − BAx = Br$ should be a better indicator of error
Right Preconditioning

Solve

\[ ABz = b \]

for \( z \). Then set \( x = Bz \).

- The preconditioned residual is the same as the original residual because \( b - A(Bz) = b - Ax \).
- The solution of the preconditioned problem is different.
- The residual may not be a good indicator of the error in \( x \).
If you can store $A$ as a sparse matrix then
- you can start a sparse factorization,
- and discard small elements in the factors,
- or enforce sparsity.

The MATLAB commands `ilu` and `ichol` create incomplete LU and Cholesky factorizations.
Integral Equations

- Many integral equations of the form
  \[ u(x) - \int k(x, y)u(y) \, dy = f(x) \] Fredholm 2\textsuperscript{nd} kind

  are well conditioned and GMRES does well.
- The transport equation is one example.
- Krylov method performance is mesh-independent.
- Preconditioning can still make a difference.
Elliptic PDEs I

Suppose you seek to solve an elliptic boundary value problem.

\[ Lu = f \]

with Dirichlet/Neumann/mixed boundary conditions.

If you discretize the PDE to obtain

\[ L_h u_h = f_h \]

the resulting discrete problem is very poorly conditioned and Krylov methods will be slow.
Split $L = L_1 + L_0$, where $L_1$ contains the high-order derivatives. If you can find a fast solver for $L_1$ with the same type of boundary conditions, then $L_1^{-1}$ is a mesh-independent preconditioner. Why? $L_1^{-1}L$ is an integral operator. (Manteufel/Parter 1990)
Example of PDE preconditioning

\[ -\nabla^2 u + c_1 u_x + c_2 u_y + c_0 u = f(x, y) \text{ for } 0 < x, y < 1 \]
\[ u(x, 0) = u(0, y) = u(x, 1) = u(1, y) = 0 \]
\[ L_1 u = -\nabla^2 u \]

Apply fast Poisson solver $N \log(N)$ work or use an efficient sparse direct solver.
Scalability

The scenario:

- Continuous problem: $Lu = f$; Discrete problem: $L_h u_h = f_h$.
- $h = 1/N$ spatial mesh width; $N^2$ number of mesh points.
- Second order accuracy: $u_h - u^* = O(h^2)$
- Preconditioner $B_h$ is “perfect”, i.e. Krylov needed to reduce error by factor of 10 is $N_k$ for all $h$.
- Cost of $B_h L_h$ matvec is $O(N)$

Then, given $h$ you can find $u_h$ up to truncation error in $O(N)$ work!
Pick $h_0 = 2^p h$ so that $L_{h_0} u_{h_0} = f_{h_0}$ is easy to solve.
Solve $L_{h_0} u_0 = f_{h_0}$

for $l=1:p$

- $h_l = h_{l-1}/2$; $u^0_l = u_{l-1}$
- Apply GMRES to $L_{h_l} u_l = f_{h_l}$ with $u^0_l$ as the start.
- Terminate when residual is reduced by factor of 10.
- Accept $u_l$

end for
Cost Analysis

- A matvec for $h_l = 2^l h$ costs $O((2^{-l} N)^2)$ operations
- We do at most $N_k$ matvecs at each level
- So...

\[
\text{Cost} \leq \sum_{l=0}^{p} N_k (2^{-l} N)^2 \leq \sum_{l=0}^{\infty} N_k (2^{-l} N)^2 \\
= N_k N \sum_{l=0}^{\infty} 4^{-l} = 4N_k N^2 / 3.
\]
The $k$th GMRES iteration is the solution of the linear least squares problem

$$
\min \| Ax - b \|
$$

where $x = \sum_{j=0}^{k-1} \gamma_j A^j r_0$

The key to a successful implementation is to solve this in an efficient and stable way.
A Questionable GMRES Implementation

How about this?

- As the iteration progresses store $A^j r_0$.
- Let $B_k = (r_0, Ar_0, \ldots A^{k-1} r_0)$
- Compute the QR factorization of $B_k = Q_k R_k$
- The $x_k = R_k^{-1} Q_k^T b$

What could go wrong?
What could go wrong?

- Accumulating $A^j r_0$ can be unstable
  Example $A = \text{diag}(1, 2, \ldots, N)$
- The cost of $B_k = Q_k R_k$ is $O(Nk^2)$.
- You have to start over with each $k$ and are not reusing the old columns.
Arnoldi Factorization is Better

Suppose one had an orthogonal projector $V_k$ onto $\mathcal{K}_k$. Then any $z \in \mathcal{K}_k$ can be written as

$$z = \sum_{l=1}^{k} y_l v_l^k$$

where $v_l^k$ is the $l$th column of $V_k$. So we can convert the problem for $x_k$ to a problem in $R^k$. Begin by writing any $x \in x_0 + \mathcal{K}_k$ as

$$x = x_0 + V_k y,$$

where $y$ is the vector of coefficients of $x - x_0$ using the columns of $V_k$ as the basis for $\mathcal{K}_k$. 
So if $x_k = x_0 + V_k y_k$ then

$$\|b - Ax_k\| = \|b - A(x_0 + V_k y_k)\|_2 = \|r_0 - AV_k y_k\|_2.$$ 

So the least squares problem for $y_k$ is

$$\min \|r_0 - AV_k y\|$$

If we can build $V_k$ in a stable way, we have solved the stability problem (but that is not completely simple). Can we do it efficiently?
The Gram-Schmidt process will

- build $V_k$ incrementally, so $V_k = (V_{k-1}, v_k)$,
- enable a fast $QR$ factorization of $AV_k$, and
- be stable (if done correctly).

Orthogonalization is the central part of the **Arnoldi** method.
The algorithm orthogonolizes each $A v_i$ against the columns of $V_k$ to construct $v_{k+1}$

$V = \text{arnoldi}(x_0, b, A, k)$

$r_0 = b - Ax_0; v_1 = r_0 / \| r_0 \|$

for $i = 1 : k$ do

$w = Av_i$

for $j = 1 : i$ do

$h_{ji} = w^T v_j (= (Av_i)^T v_j); w = w - h_{ji} v_j$

end for

$h_{ki} = \| w \|; v_{i+1} = w / h_{ki}$

end for
Examine the Arnoldi Loops

What if you divide by zero in

\[ v_1 = r_0 / \| r_0 \| \text{ or } v_{i+1} = w / \| w \| ? \]

- If \( r_0 = 0 \), then \( x_0 \) is the solution and the GMRES iteration would terminate.
- If \( w = 0 \), then you have a happy breakdown of the Arnoldi process. This implies that you found the solution as \( x_{k-1} \).
- A well-designed implementation would stop before division by zero.
Results from Arnoldi

- \( V_{k+1} \): orthonormal basis for \( \mathcal{K}_k \).
- \( H_k \): \((k + 1) \times k\) upper Hessenberg matrix

and the Arnoldi factorization

\[
AV_k = V_{k+1}H_k
\]

So, setting \( \beta = \|r_0\|_2 \).

\[
\|r_0 - AV_k y\|_2 = \|r_0 - AV_k y\|_2 = \|r_0 - V_{k+1}H_k y\|_2
\]

\[
= \|\beta e_1 - H_k y\|_2
\]
Why Arnoldi is good

We have converted the $N \times k$ least squares problem for the GMRES iterate $x_k$ to

$$\min \| \beta e_1 - H_k y \|_2$$

which is $(k + 1) \times k$ and $H_k$ is upper Hessenberg.

- So we can solve it with Givens rotations.
- When done cleverly, it’s $O(k^2)$ work/iteration to keep the QR factorization of $A$ up-to-date and compute $y$. 
- I did the orthogonalization with CGS. That’s not very good.
  - Use CGS2 and make sure that $v_{k+1}$ and the $k$th column of $H$ come out correctly.
- Do the Givens rotations astutely.
- We never compute $\|r\|$ directly.
A Framework for GMRES Implementation

\[ r = b - Ax, \quad v_1 = \frac{r}{\|r\|_2}, \quad \rho = \|r\|_2, \quad \beta = \rho, \quad k = 0 \]

while \( \rho > \epsilon \|b\|_2 \) and \( k < k_{\text{max}} \) do

\[ k = k + 1 \]

Apply Arnoldi (orthogonalization!)

to obtain \( H_k \) and \( V_{k+1} \) from \( V_k \) and \( H_{k-1} \)

\[ e_1 = (1, 0, \ldots, 0)^T \in \mathbb{R}^{k+1} \]

Solve \( \min ||\beta e_1 - H_k y_k||_{R^{k+1}} \) for \( y_k \in \mathbb{R}^k \).

\[ \rho = ||\beta e_1 - H_k y_k||_{R^{k+1}}. \]

end while

\[ x_k = x_0 + V_k y_k. \]
What you need to remember.

- Do not write your own GMRES code! Use tools!
- Costs are dominated by
  - The matrix-vector products $\mathbf{Av}$
  - The orthogonalization of $\mathbf{v}_{k+1}$ against columns of $\mathbf{V}_k$
  - Work and storage increase with $k$.
- Storage increases with $k$. This can be very bad.
- Poor orthogonalization means
  - $\rho = \| \beta \mathbf{e}_1 - \mathbf{H}_k \mathbf{y}_k \|$ is a poor approximation to $\mathbf{r}_k$
  - So you’ll terminate with incorrect results.
Bad things happen with loss of orthogonality.

You knew this, but I will tell you again. Take this problem, please.

\[
A = \begin{pmatrix}
0.001 & 0 & 0 \\
0 & 0.0011 & 0 \\
0 & 0 & 10^4
\end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad \text{and} \quad \mathbf{x}_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}
\]

We’ll solve \( \mathbf{A}\mathbf{x} = \mathbf{B} \) with GMRES. We should converge in 3 iterations.
## Trying different orthogonalizations: \( \| r_k \| / \| r_0 \| \)

<table>
<thead>
<tr>
<th>( k )</th>
<th>CGS</th>
<th>MGS</th>
<th>MGS2</th>
<th>CGS2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.00e+00</td>
<td>1.00e+00</td>
<td>1.00e+00</td>
<td>1.00e+00</td>
</tr>
<tr>
<td>1</td>
<td>8.16e-01</td>
<td>8.16e-01</td>
<td>8.16e-01</td>
<td>8.16e-01</td>
</tr>
<tr>
<td>2</td>
<td>3.88e-02</td>
<td>3.88e-02</td>
<td>3.88e-02</td>
<td>3.88e-02</td>
</tr>
<tr>
<td>3</td>
<td>6.69e-05</td>
<td>6.42e-08</td>
<td>6.42e-08</td>
<td>6.34e-34</td>
</tr>
<tr>
<td>4</td>
<td>4.74e-05</td>
<td>3.70e-08</td>
<td>5.04e-24</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>3.87e-05</td>
<td>3.04e-18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>3.35e-05</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>3.00e-05</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>2.74e-05</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>2.53e-05</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>2.37e-05</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
An Evil Matrix for GMRES

\[
J = \begin{pmatrix}
1 & 1 & 0 \\
0 & 1 & 1 & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 1 & 1 & 0 \\
0 & 1 & 1 \\
0 & 1
\end{pmatrix}, \quad b = \begin{pmatrix}
1 \\
0 \\
\vdots \\
0 \\
0 \\
0
\end{pmatrix}, \quad x_0 = \begin{pmatrix}
0 \\
0 \\
\vdots \\
0 \\
0 \\
1
\end{pmatrix},
\]

Solution: \( x^* = b \) is an eigenvector of \( A \).
\( \lambda = 1 \) is the only eigenvalue. Nice clustering, but ...
Solution of $Jx = b$ with GMRES: $N = 4$

What happened?

<table>
<thead>
<tr>
<th>$k$</th>
<th>$|r_k|$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3.16e+00</td>
</tr>
<tr>
<td>1</td>
<td>8.62e-01</td>
</tr>
<tr>
<td>2</td>
<td>2.35e-01</td>
</tr>
<tr>
<td>3</td>
<td>1.19e-01</td>
</tr>
<tr>
<td>4</td>
<td>7.94e-32</td>
</tr>
</tbody>
</table>
J is the world’s most Krylov-hostile matrix.

- It’s a Jordan block, which has nothing to do with the NBA.
- $N \times N$ Jordan blocks have eigenvalues with
  - algebraic multiplicity $N$,
  - geometric multiplicity 1.
- You can’t get more non-diagonalizable than that.

So the theorems are true, but Jordan blocks can bite you.
Bottom line on GMRES: theory

- GMRES is happy if
  - \( A \) is diagonalizable,
  - \( \kappa(V) \) is not too large,
  - and the eigenvalues are clustered.

- GMRES can be very unhappy if
  - \( A \) is not diagonalizable,
  - \( \kappa(V) \) is large,
  - \( \kappa(A) \) is large.

- You can fix large \( \kappa(A) \) with preconditioning.
Boundary value problems need preconditioning.

Integral equations generally don’t.

Neither of these lead to Jordan blocks that cause problems, if you do your preconditioning job right.
Bottom line on GMRES: implementation

- Mat-vecs and orthogonalization are the expensive parts.
- CGS2 is a sensible way to go on machines with two or more cores.
- MGS2 is ok for single core machines (phones, pagers, ...)
- Storage can be a serious problem. More on this later.