Some notes about Iterative methods for solve linear equations

(Konstantin Burlachenko, 15 OCT 2018, 13DEC2018 burlachenkok@gmail.com , bruziuz@stanford.edu)

1. References

[CS193G] CS193G Guest lecture about Matrix-Vector multiplication, 2010

[MIT] MIT, Prof. Gilbert Strang, Mathematical Methods for Engineers II

https://ocw.mit.edu/courses/mathematics/18-086-mathematical-methods-for-engineers-ii-spring-2006/readings/

[SAMARSKIY] Численные методы А.А.Самарский, А.В.Гулин, 1989

(Samarskiy A.A., Gulin A.V. Chislennie metody)

(http://samarskii.ru/books/book1989.pdf)

[CS205A] Mathematical Methods for Robotics, Vision, and Graphics, http://graphics.stanford.edu/courses/cs205a/schedule.html

[EE364B] Convex Optimization II http://web.stanford.edu/class/ee364b/

1. Introduction

Solve system of linear equation based on factor-solve methods like factorization based on Cholesky decomposition (if matrix A is p.d.) or Gauss elimination (if matrix A is arbitrarily) directly o matrix A for Ax=b for big matrices is slow and impractical for extremely big problem because algorithm complexity grows as n^3, where n is number of rows and number of columns in matrix A for solve $Ax = b$.

What can potentially help with big problems:

1. Use iterative methods. But they are not so universal and they can "not converge".
2. Use sparse factorization. But there are still selection "ordering" heuristic in them. And probably matrix factor will have big *fill-in* factor during factorization matrix A due to such heuristic nature of methods.

This is a note about couple powerful iterative methods. But firstly small note about sparse numerical methods.

2. Problems with sparse numerical methods

Fill-In – is the situation when you work with matrix in some way with some algorithm and original sparsity properties of the matrix are losing. Assume we need solve system of linear equations $Ax = b$. If use just *Gauss Elimination* for dense matrices then a lot of Fill-In is start to going on.

For sparse matrix we can fast perform matrix-vector product. But elimination even with good ordering can be rather expensive or in terms of time or in terms of storing. One way to overcome this is use iterative methods for solving $Ax = b$.

Especially benefit is that it's *"any time stop"* methods as I heard during [EE364B] lectures with S.Boyd. For any iterative method specific preconditioning can be applied.

In lections [CS193G] I heard that unfortunately *"Gasuss-Seidlel"* is not so well map to CUDA, because it's not so parallel compare to Jacobi. But there are approximations which do approximately Gauss-Seidel which is called: ***Red-Black Gauss-Seidel relaxation***.

3. Several ways to solve system of linear equation in iterative fashion

In lectures from [MIT] there it have been mentioned that problem can be attacked from several directions. And there are:

1. Iteration methods

2. Multigrid methods

3. Krylov methods (Mostly famous is Conjugate Gradients method for symmetric and p.d. A)

4. Iteration methods for solve system of linear equation

This material is obtained from lectures on iterative methods from [MIT].

$$Ax = b \leq > 0 = b - Ax \leq > Px = b + PIx - Ax = b + (PI - A)x$$

Now we use this equation to construct iterative schema: $Px_{k+1} = (P - A)x_k + b$

If it will converge then it will satisfy equation above and it with non-singular P it will automatically be an answer for $Ax = b$

If $P = A$ it's one extreme. But it will lead to problem solve $Px_{k+1} = Ax_{k+1} = b$

Possible choices:

A. Set $P_1 = diag(A)$ it's called **Jacobi** method.
B. **P = D/w** . It one modification take scalar multiple of diag(A) it's called ***Weighted Jacobi***.
C. Set $P_2 = lowerTrinagularPart(A)$ including diagonal of A. it's called **Gauss-Seidel** method.
D. Set P_3 of combination of P_1 and P_2. Successive over relaxation (SOR).
E. Set $P_3 \approx LU$. ILU. Incomple LU. Idea of this do LU but don't allow fill-in. Completely ignore fill-in or allow some percentage of total size.

4.1 Analyze of convergence of iterative method

Now we can perform analyze of convergence in the following way:

$Px_{k+1} = (P - A)x_k + b$ (1)

$Px_{true} = (P - A)x_{true} + b$ (2)

Let's say: $e_k = x_k - x_{true}$.

Let's evaluate (1)-(2). Its gives: $Pe_{k+1} = (P - A)e_k$ =>

$$\boxed{e_{k+1} = \left(I - P^{-1}A\right)e_k}$$

It's called **error equation**

Iteratively repeat this equation for new iterates we will obtain: $e_k = \left(I - P^{-1}A\right)^k e_0$

Several Observations about iterative method in such schema:

1. If all $\left|\lambda_i\left(I - P^{-1}A\right)\right| < 1$ then algorithm converge for any e_0
2. $\left|\lambda\left(I - P^{-1}A\right)\right|$ is called spectral radius of $M = \left(I - P^{-1}A\right)$
3. Gauss-Seidel use new information
4. Jacobi use old information.

Roughly speaking one iteration of Gauss-Seidel is as two of Jacobi steps.
https://www.youtube.com/watch?time_continue=3023&v=LtNVodIs1dI
(Lecture 15 | MIT 18.086 Mathematical Methods for Engineers II)

5. Idea in some way choose easy invertible matrix P which lead to small spectral radius of $\left(I - P^{-1}A\right)$
6. First step in all iterative code – reordering with guarantee that diagonal is not zero

4.2. Several Theorems relative to iterative methods from Russian Math Books

Theorem of A.A. Samarskiy

A.A.Samarskiy use another form of canonical form of iterative method, rather in lectures from [MIT]. Iterative one-step stationary method in canonical form is described by:

$B\frac{x_{n+1}-x_n}{\tau} + Ax_n = b$ and $det(B)\neq 0$ [*]

Stationary in this context means B is constant matrix and τ is constant scalar, which does not depend on iteration.

Theorem: If A is p.d. and $\tau > 0$ and $B - \frac{1}{2}\tau A > 0$ then iterative procedure is converge

Consequences of this Theorem mentioned in [SAMARSKIY]:

1. Jacobi method for symmetric and p.d. matrix A, which has diagonal dominance will converge
2. If A is p.d. then Gauss-Seidel method will converge
3. If A is not symmetric then if all eigenvalues of $T = E - \tau B^{-1}A$ by norm is less then 1. Then iterative procedure [*] converge for any x_0
4. If B is not identity and A is symmetric and B is symmetric then optimal τ_0 is:

$$\tau_0 = \frac{2}{\lambda_{max}(B^{-1}A)+\lambda_{min}(B^{-1}A)}$$

In book it have been shown that error in each iteration has following property: $\left|e_N\right|_2 \leq \rho^N \left|e_0\right|_2$

Where

$$\rho = \frac{\lambda_{max}(B^{-1}A)-\lambda_{min}(B^{-1}A)}{\lambda_{max}(B^{-1}A)+\lambda_{min}(B^{-1}A)} = \frac{\frac{\lambda_{max}(B^{-1}A)}{\lambda_{min}(B^{-1}A)}-1}{\frac{\lambda_{max}(B^{-1}A)}{\lambda_{min}(B^{-1}A)}+1}$$

$\frac{\lambda_{max}(B^{-1}A)}{\lambda_{min}(B^{-1}A)}$ for symmetric and p.d. matrix $B^{-1}A$ is **condition number**.

4.3 Important consequence for iterative methods

Conclusion.
When the system is ill-conditioned then iterative method works very poor because ρ will be just only slightly less then 1. Art and tweaking is find B such that condition number of B is fine enough.

Even idea behind iterative methods sounds great, but it looks that iterative methods are not so universal as for example factor-based methods.

5. Multigrid methods

5.1 Multigrid method description

https://cosmolearning.org/video-lectures/sparse-systems/

I saw videos about this method from [MIT]. Authors are (William L. Briggs, and others).

Idea that we have problem $A_h u_h = b_h$ on fine grid.

1. First idea is to use Jacobi or Gauss-Seidel for several iteration on whole grid. But do it only 3 or 4 times to get rough solution. Compute residual $r_h = b_h - A_h u_h$
2. Go to coarse grid via restriction matrix $r_{2h} = R_{[2h,h]} \bullet r_h$. One possible solution is just take items felled into coarse grid, another option take neighbors into account.
3. Now we have smaller problem
4. "Solve" problem (probably recursively) in coarse grid for residual $A_{2h} E_{2h} = r_{2h}$
    a. The problem is that we should someway create this A_{2h}
5. Interpolate backward $E_h = I_{[h,2h]} \bullet E_{2h}$
6. Add E_h to u_h and now we should have better answer

I – is interpolation in 1D. One possible choice is rectangular matrix filled with 0,1,1/2
R – is restriction/ down sampling in 1D.
One possible choice $R = \frac{1}{2} I^T$

It seems that for grids to be consistent the following should be hold: $A_{2h} = R A_h I$
I.e. effect of apply linear operator A_{2h} is the same as make up sampling, apply operator in more fine grid and then down sample the result.

With such structure it's possible to get good answer. And if size of (A) is $n = N^2$ then multigrid works in *O(n)* flops.

5.1 Multigrid method analysis

1-st step - is doing via iteration and matrix for iteration is $(E - P^{-1} A)$ (From section 3 of this document)

2-6 steps - can be done in fact be fused into via one matrix multiplication

Error in formed solution is $e_h = u - u_h$

We assume that we constructed grid such that for true solution we have: $A_h u = b_h$

So $r_h = b_h - A_h u_h = A_h u - A_h u_h = A_h e_h$

For step#2 we have $r_{2h} = Rr_h = RA_h e_h$

For step#4 we have $A_{2h}E_{2h} = r_{2h} = RA_h e_h$

=> $E_{2h} = A_{2h}^{-1} RA_h e_h$

For step#5 we have

=> $E_h = I \cdot E_{2h} = IA_{2h}^{-1} RA_h e_h = (IA_{2h}^{-1} RA_h) e_h = Se_h$

For step #6 we have: add E_h to u_h

$u_h^{'} = u_h + E_h = u_h + Se_h$

We can massage it into following form:

$e_h = u - u_h$

$u - u_h^{'} = u - u_h - Se_h$ =>

$e_h^{'} = (Identity - S)e_h$
$S = IA_{2h}^{-1} RA_h$
But also because $A_{2h} = RA_h I$ =>
$S = I\left(RA_h I\right)^{-1} RA_h$
$(Identity - S)$ called as multigrid matrix.

5.2 Nice property of eigenvalues for S such that $S^2 = SS$

$$S^2 = SS = I\left(RA_h I\right)^{-1} RA_h \cdot I\left(RA_h I\right)^{-1} RA_h = I\left(RA_h I\right)^{-1}\left(RA_h I\right)\left(RA_h I\right)^{-1} RA_h = I\left(RA_h I\right)^{-1} RA_h = S$$

So for matrix S we have $S^2 = S$. By the way it doesn't mean that ***S*** have to be symmetric!

If $S^2 = S$ then only possible eigenvalues are 0,1.

Prove: If $Se = \lambda e$ [1]

$S(Se) = S\lambda e = \lambda(Se)$ [2]

⇨ $Se = \lambda(Se)$ (because $S^2 = S$)

Now use equation [1] for last equation

$\lambda e = \lambda(\lambda e)$ <=> $\lambda^2 e - \lambda e = 0$ and $e \neq 0$ (because it is eigenvector and eigenvectors are non-zero)

And so $\lambda_j = 0$ or $\lambda_j = 1$

So eigenvalues for this type of matrices are real and moreover can have only two possible values.

5.3 Conclusion of the analysis

$e_h^{'} = (Identity - S)e_h$

$(Identity - S)e_h$ affect into ways for error.

If e_h lie in span eigenvalues equal to 1 then multigrid completely remove it.

If e_h lie in span eigenvalues equal to 0 then multigrid step do nothing.

S has size [n,n] and has eigenvalues equal to 1 and to 0.

6. Conjugate Gradients based method

Material for this section have been taken from [CS205A], [EE364B].

First of all **good news**. It's amazing algorithm for solve system with p.d. matrix A.

In this case problem $Ax = b$ is the same as problem $min.(\frac{1}{2}x^T Ax - b^T x + c)$

I heard about two interpretations of this algorithm.

6.1. First interpretation of CG

Useful reference on the subject is:

J.R.Shewchuk, An Introduction to the Conjugate Gradient Method without Agonizing Pain, 1995

We can try to find minimum of this function via gradient descend. So we select a direction. Answering for question "*How long step should be done?*" we do it in greedy way and perform *exact step length* which minimize function over the ray.

$$d_k\left(x_{k-1}\right) = -\ grad\left(\left(\frac{1}{2}x^T Ax - b^T x + c\right)\right) = b - Ax_{k-1}$$

It can be shown that fir exact line search: $\alpha_k = \frac{d_k^T d_k}{d_k^T A d_k}$

$$x_k = x_{k-1} + \alpha_k d_k$$

Speed for gradient descent slow generally speaking in can be shown that:

$$\left|e_i\right|_A = \left|x - x^*\right|_A \leq \left(\frac{k-1}{k+1}\right)^i \left|x_0 - x_*\right|_A = \left(\frac{k-1}{k+1}\right)^i \left|e_0\right|_A$$

$$k = cond(A) = \frac{\lambda_{max}(A)}{\lambda_{min}(A)}$$

And $\left|e_i\right|_A$ means A-quadratic norm defined as: $\left|e_i\right|_A = \left|e_i^T A e_i\right|^{\frac{1}{2}}$

So gradient descend works great for matrix of quadratic form has condition number near to 1.

Idea of CG is to change coordinates such that in new variables the problem is more easy solved.

Instead solve $min._x(\frac{1}{2}x^T Ax - b^T x + c)$ we **assume** that we know Cholesky factorization.

So we have $min._x(\frac{1}{2}x^T LL^T x - b^T x + c)$ but if know change coordinates into $y = L^T x$ then in new coordinates this quadratic function will have spherical sublevel sets.

And in new space gradient descend will be worked very quickly in fact gradient descend will converge in one step.

If think deeply then take gradient descend method in ***distorted space*** is the same as select direction in original space, by select special transformation. Directions should be A-orthogonal because $\left(L^T v\right)^T L^T w = 0 <=> v^T Aw = 0$

If work in ***distorted space*** then if we select n mutually-orthogonal directions then it will be guaranteed that we will converge in ***n*** steps because quadratic form is identical matrix.

Conceptually if you have some directions in distorted space then any orthogonalization in distorted space will work. And it can be done for examples *Gramm-Schmidt*

Algorithm:

1. $v_k = r_{k-1} - \frac{\left(r_{k-1}, v_{k-1}\right)_A}{\left(v_{k-1}, v_{k-1}\right)_A}\left(v_{k-1}\right)$
2. $\alpha_k = \frac{v_k^T r_{k-1}}{v_k A v_k}$
3. $x_k = x_{k-1} + \alpha_k v_k$
4. $r_k = r_{k-1} - \alpha_k A v_k$

Properties of algorithm:

1. Converge in n steps to solution
2. There are also different forms
3. Convergence speed for CG is $\left|e_i\right|_A \leq \left(\frac{\sqrt{k}-1}{\sqrt{k}+1}\right)^i \left|e_0\right|_A$
4. Just as a reminder for Gradient descend Convergence speed is: $\left|e_i\right|_A \leq \left(\frac{k-1}{k+1}\right)^i \left|e_0\right|_A$

6.2 Second interpretation of CG

From [EE364B].

$K_k = span\{b, Ab, \ldots, A^{k-1}b\}$ is *Krylov* subspace.

$x_k = argmin_{x \epsilon K_k}(f(x)) = argmin_{x \epsilon K_k}\left(\left|x - x^*\right|_A^2\right)$ is Krylov sequence of points.

$f\left(x_{k+1}\right) \leq f(x_k)$ (because we increase domain of objective in each iteration)

$f\left(x_n\right) = x^*$ (Follow from Cally - Hamilton Theorem from Linear Algebra)

What is great and not obvious there are a two term recurrences how calculate x_{k+1} from x_k, x_{k-1}.

As S.Boyd showed in fact we can at ***k*** iterate just create polynomial ***p*** such that $p_k\left(\lambda_i\right) = \frac{1}{\lambda_i}$ where λ_i are eigenvalues of *A*. Or it can reformulated in this way:

If there is a polynomial *q* of degree k such that q(0)=1 and q is small on position of eigenvalues of A then we're done.

Pluses:

1. Convergence speed can be very fast if there is a small number of cluster of eigenvalues.
2. If eigenvalues clustered in k groups then it's possible to have degree k polynomial which is pretty small on center of clusters.
3. If there only two unique eigenvalues then second iteration of CG gives answer for $x = A^{-1}b$.
4. Can give approximate solution in few steps
5. Interesting applications of CG you even can not store explicitly matrix A.
6. There is a two term recurrence for evaluate next iteratate in Krylov sequence.
7. If solution is approximately combination of k eigenvectors then k-th iteration of CG gives good approximate solution
8. The main plus is in that t**here is no need to store A**, so if A is sparse and huge it will be in room of this method.
9. Also this method support warm start. Common schema if use for Newton method – if previous direction always a descend direction then use it as warm start.
10. Can be used to solve extremely high problems
11. Applied to Newton Method called BFGS or iterative Newton method relative to limited memory
12. CG for [9] is something between gradient descent (if take one step) and Newton method (if make N steps with exact arithmetic)
13. Every CG step makes angle from current direction more tight for newton direction.
14. Speed up relative to Newton method can be ~x1000
15. CG can diverge it's better to collect history of all errors and pickup solution with the smallest error
16. Common approach during iterative fashion. Take dx from previous step. If it's descend direction then use it. If not initialize with zero.

Minuses:

1. Works for symmetric p.d. A
2. Due to round-off errors works poorly or not at all
3. Less reliable then Newton method

4. Residual can go up
5. *S.Boyd: "Theory is useless, we don't want to run CG for n steps"*, we want run it for 10 steps.

Various material relative to CG from EE364B class by S.Boyd:

Link	Description
https://youtu.be/E4gl91l0l40?t=4371	Diagonal preconditioning CG is very very cool
https://youtu.be/E4gl91l0l40?t=3494	Truncated Newton Method
https://youtu.be/E4gl91l0l40?t=1617	Efficient MV multiply. (FFT, etc.)
https://youtu.be/E4gl91l0l40?t=3157	One of the form of Limited Memory BFGS