AMATH 585 Assignment 7

Tyler Chen

Problem 1

The conjugate gradient method for solving a symmetric positive definite linear system $Ax = b$ can be written as below:

```
Given $x_0$, compute $r_0 = b - Ax_0$, and set $p_0 = r_0$.
For $k = 1, 2, \ldots,$
   Compute $Ap_{k-1}$.
   Set $x_k = x_{k-1} + a_{k-1}p_{k-1}$, where $a_{k-1} = \frac{\langle r_{k-1}, r_{k-1}\rangle}{\langle p_{k-1}, Ap_{k-1}\rangle}$.
   Compute $r_k = r_{k-1} - a_{k-1}Ap_{k-1}$.
   Set $p_k = r_k + b_{k-1}p_{k-1}$, where $b_{k-1} = \frac{\langle r_k, r_k\rangle}{\langle r_{k-1}, r_{k-1}\rangle}$.
Endfor
```

(a) Show that the residual vectors $r_0, \ldots, r_k$ are orthogonal to each other ($\langle r_i, r_j\rangle = 0$ if $i \neq j$) and that the direction vectors $p_0, \ldots, p_k$ are A-orthogonal ($\langle p_i, Ap_j\rangle = 0$ if $i \neq j$). [Hint: First show that $\langle r_1, r_0\rangle = \langle p_1, Ap_0\rangle = 0$ and then use induction on k.]

(b) If A is the $N \times N$ matrix of the 5-point operator for Poisson's equation on a square, count the number of operations (additions, subtractions, multiplications, and divisions) performed in each iteration. (Show how you arrive at your result.)

(c) Compare your operation count in (b) to that of a Gauss-Seidel iteration applied to the same N by N 5-point matrix A. Also compare to the operation count for a multigrid V-cycle, using one Gauss-Seidel iteration on each visit to each grid level. (Again, show how you derive your operation counts.)

Solution

(a) Note that $p_0 = r_0$ and A is SPD (implies self adjoint) so that $\langle x, Ay\rangle = \langle Ax, y\rangle$. Then,

$$a_0 = \frac{\langle r_0, r_0\rangle}{\langle p_0, Ap_0\rangle} = \frac{\langle r_0, r_0\rangle}{\langle p_0, Ap_0\rangle}$$

We have,

$$\langle r_1, r_0\rangle = \langle r_0 - a_0Ap_0, r_0\rangle = \langle r_0, r_0\rangle - \frac{\langle r_0, r_0\rangle}{\langle p_0, Ap_0\rangle}\langle Ar_0, r_0\rangle = 0$$

Likewise, using this result and the fact that $b_0 = \langle r_1, r_1\rangle / \langle r_0, r_0\rangle$,

$$\langle p_1, Ap_0\rangle = \langle r_1 + b_0p_0, (r_0 + r_1)/a_0\rangle = (\langle r_1, r_0\rangle + \langle r_1, r_1\rangle + b_0(\langle r_0, r_0\rangle + \langle r_0, r_1\rangle))/a_0 = 0$$

Suppose $\langle r_i, r_j\rangle = \langle p_i, Ap_j\rangle$ for all $i, j \leq k$. Then, for $j < k$ we have,

$$\langle Ap_k, r_j\rangle = \langle Ap_k, p_j - b_{j-1}p_{j-1}\rangle = \langle Ap_k, p_j\rangle - b_{j-1}\langle Ap_k, p_{j-1}\rangle = 0$$

Therefore, for $j < k$ we have,

$$\langle r_{k+1}, r_j\rangle = \langle r_k - a_kAp_k, r_j\rangle = \langle r_k, r_j\rangle - a_k\langle Ap_k, r_j\rangle = 0$$

If $j = k$ then, $a_k = \langle r_k, r_k \rangle / \langle p_k, Ap_k \rangle$ so,

$$\langle r_{k+1}, r_k \rangle = \langle r_k - a_k A p_k, r_k \rangle = \langle r_k, r_k \rangle - a_k \langle Ap_k, p_k \rangle = 0$$

Using this fact, for $j < k$, we now have,

$$\langle r_{k+1}, Ap_j \rangle = \langle r_{k+1}, (r_j - r_{j+1})/a_j \rangle = \langle r_{k+1}, r_j \rangle / a_j - \langle r_{k+1}, r_{j+1} \rangle / a_j = 0$$

Therefore, for $j < k$ we have,

$$\langle p_{k+1}, Ap_j \rangle = \langle r_{k+1} + b_k p_k, Ap_j \rangle = \langle r_{k+1}, Ap_j \rangle + b_k \langle p_k, Ap_j \rangle = 0$$

If $j = k$ then, since $1/a_k = \langle p_k, Ap_k \rangle / \langle r_k, r_k \rangle$ and $b_k = \langle r_{k+1}, r_{k+1} \rangle / \langle r_k, r_k \rangle$,

$$\begin{aligned}\langle p_{k+1}, Ap_k \rangle &= \langle r_{k+1} + b_k p_k, Ap_k \rangle \\ &= \langle r_{k+1}, (r_k - r_{k+1})/a_k \rangle + b_k \langle p_k, Ap_k \rangle \\ &= \langle r_{k+1}, r_k \rangle / a_k - \langle r_{k+1}, r_{k+1} \rangle / a_k + b_k \langle p_k, Ap_k \rangle = 0\end{aligned}$$

Given the base case proven earlier this proves $\langle r_i, r_j \rangle = \langle p_i, Ap_j \rangle = 0$ for all $i \neq j$. □

(b) Let $A \in \mathbb{R}^{n\times n}$. Then $x_k, r_k, p_k \in \mathbb{R}^n$. Computing a vector sum and scalar multiply each take n operations. Computing in inner product takes $2n - 1$ floating point operations. If A is the matrix from the 5-point Poisson operator it has at most 5 entries in a given row. We assume that n is large so that the corners and edge cases do not matter. Therefore multiplying a single row of A by a vector will take 5 multiplications and 4 additions. We do this for all n rows. Therefore computing the product of A with a vector takes $9n$ operations.

The leading order term will be n. We therefore do not count the divisions in the coefficients.

In the first line we have one matrix vector product.

In the second line we have two inner products, one vector add, and on scalar multiply. However, one inner product was computed earlier, so we can skip this if we store it in memory.

In the third line one vector add, and on scalar multiply.

In the fourth line we have two inner product, one vector add, and one scalar multiply. However, again one inner product was computed earlier.

In total we have, one matrix vector product, two inner products, three vector additions, three scalar multiplies. This gives,

$$\Theta(9n + 2(2n) + 3(n) + 3(n)) = \Theta(19n) \text{ operations}$$

(c) For a Gauss-Seidel step we first compute the residual $b - Ax$ (one matrix vector product and one vector subtract).

We then solve $Me = r$. In GS iteration M is the lower triangular part of A. Therefore there are at most 3 nonzero entries per row. Using back substitution, in each row we solve something like,

$$ax + by + cz = r \qquad \Longleftrightarrow \qquad z = (r - ax - by)/c$$

This is two multiplications, two subtractions and one division. Therefore for the whole matrix we have $5n$ operations.

Finally, set $x = x + e$ (one vector add).

In total we have, one matrix vector product, one vector subtract, one sparse solve using back substitution, and one vector add. This gives,

$$\Theta(9n + n + 5n + n) = \Theta(16n) \text{ operations}$$

Suppose we have a multi-grid V-cycle. At each level we do a Gauss-Seidel step, project, (do the iterative part), interpolate back, and do another Gauss-Seidel step. We assume a linear interpolation matrix, and the natural projection matrix.

We will come up with an upper bound for the leading coefficient of the number of operations.

We showed a Gauss-Seidel step takes $\Theta(16n)$ operations.

Our interpolation matrix averages either 1 point (if the point is in the coarse grid), 2 points (if it is between two points in the coarse gird), or 4 points (if the point is between 4 points in the coarse grid). In the worse case we average 4 points together (3 additions one division). This occurs for $\mathcal{O}(n/4)$ points.

Projecting down takes constant time since we do not do any floating point operations (although we do read and write to memory $\mathcal{O}(n)$ times.

Interpolating takes less than 9 multiplications and 8 additions for each of the $(n-1)/2$ rows.

At one level we then have,

$$\Theta(16n + 4(n/4) + 0 + 16n) = \Theta(33n) \text{ operations}$$

Suppose we make k steps until we can solve the final system easily (in $\mathcal{O}(1)$. At each successive grid level the size of the matrices decreases by a factor of two for each dimension for a total factor of 4. Then the algorithm takes,

$$\sum_{k=0}^{n} \left(\frac{1}{4}\right)^k \Theta(33n) \text{ operations}$$

In the limit $k \to \infty$ we have,

$$(4/3)\Theta(33n) = \Theta(44n) \text{ operations}$$

We note that reading and writing to the memory takes non-negligible time, especially for sparse array computations. Some things like scalar multiplications might not need to be computed at each stage either depending on the algorithm. In short, these computations are a good approximation to the actual result, but floating point operations are not the single most important thing, especially for large systems on large computers.

Problem 2

Implement a 2-grid method for solving the 1D model problem with homogeneous Dirichlet boundary conditions:

$$u_{xx} = f(x), \quad u(0) = u(1) = 0.$$

Use linear interpolation to go from the coarse grid with spacing $2h$ to the fine grid with spacing h. Take the projection matrix I_h^{2h}, going from the fine grid to the coarse grid, to be 0.5 times the transpose of the interpolation matrix: $I_h^{2h} = \frac{1}{2}(I_{2h}^h)^T$. Use a multigrid V-cycle with 1 smoothing step on each visit to each grid level. Try weighted Jacobi and Gauss-Seidel as the smoothing step. Try several different values of the mesh spacing h and show that you achieve convergence to a fixed tolerance in a number of cycles that is independent of the mesh size.

Solution

We implement a general arbitrary depth V-cycle multigrid solver.

```
def multi_grid(A_,f_,M_,u0,J,max_iter,tol):
    ns = [len(u0)]
    for j in range(J):
        ns.append(int((ns[-1]+1)/2)-1)

    al = u0[0]
    be = u0[-1]
    def b_(f,n,al,be):
        b = f
        b[0] -= al*(n+1)**2
        b[-1] -= be*(n+1)**2

        return b

    A = list(map(A_,ns))
    M = list(map(M_,A))
    xs = list(map(lambda n:np.linspace(0,1,n+2)[1:-1],ns))
    b = b_(f_(xs[0],0.5),ns[0],al,be)

    Ifc = list(map(Ifc_,ns))[1:]
    Icf = list(map(Icf_,ns))

    r = list(map(lambda n:np.zeros(n),ns))
    f = list(map(lambda n:np.zeros(n),ns))
    delta = list(map(lambda n:np.zeros(n),ns))
    d = list(map(lambda n:np.zeros(n),ns))

    u=u0
    res_norm = []
    r[0] = b - A[0].dot(u)
    for k in range(max_iter):
        res_norm.append(np.linalg.norm(r[0])/np.linalg.norm(b))
        if res_norm[-1] < tol:
            return u,k

        # smooth
        for j in range(1,J):
            f[j] = Ifc[j-1].dot(r[j-1])
            delta[j] = sparse.linalg.spsolve_triangular(M[j],f[j])
            r[j] = f[j] - A[j].dot(delta[j])

        # project
        f[J] = Ifc[J-1].dot(r[J-1])

        # solve on corse grid
        d[J]= sparse.linalg.spsolve(A[J],f[J])

        # smooth
        for j in range(J-1,0,-1):
            delta[j] += Icf[j+1].dot(d[j+1])
            d[j] = delta[j] + sparse.linalg.spsolve_triangular(M[j],f[j]-A[j
                ].dot(delta[j]))

        # interpolate
        u += Icf[1].dot(d[1])

        r[0] = b - A[0].dot(u)
        u += sparse.linalg.spsolve_triangular(M[0],r[0])

    return u,-1
```

We the solver with $J = 1$ so that there are two grid levels. A sample solution is shown in Figure 1 after the residual has reached a tolerance of 1^{-10}.

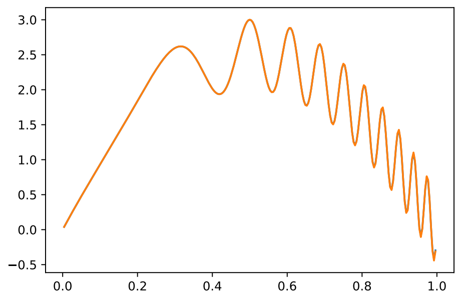
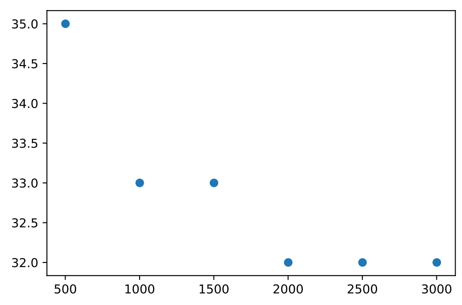
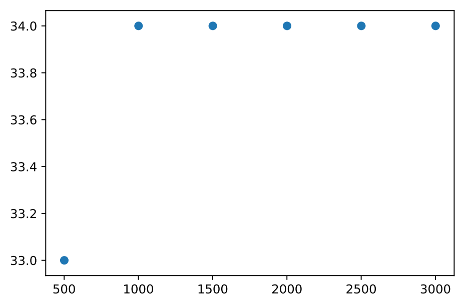


Figure 1: Sample solution (orange) vs. actual solution (blue)

We use a direct solve on the coarse grid, and one simple iteration smooth on the fine grid. Figure 2 shows the number of iterations required for the residual to reach a tolerance of 1^{-10} using weighted Jacobi and Gauss-Seidel in the smoothing step. Note that the number of iterations does not change appreciably as the mesh size increases.



(a) weighted Jacobi ($\omega = 2/3$)



(b) Gauss-Seidel

Figure 2: Number of iterations for residual to reach 1^{-10}

Problem 3

(a) Consider an iteration of the form

$$x_k = x_{k-1} + M^{-1}(b - Ax_{k-1}),$$

for solving a nonsingular linear system $Ax = b$. Note that the error $e_k := A^{-1}b - x_k$ satisfies

$$e_k = (I - M^{-1}A)e_{k-1} = \ldots = (I - M^{-1}A)^k e_0.$$

Assume that $\|e_0\|_2 = 1$ and that $\|I - M^{-1}A\|_2 = \frac{1}{2}$. Estimate the number of iterations required to reduce the 2-norm of the error below 2^{-20}. Show how you obtain your estimate. Now suppose you know only that the spectral radius $\rho(I - M^{-1}A) = \frac{1}{2}$. Can you give an estimate of the number of iterations required to reduce the 2-norm of the error below 2^{-20}? Explain why or why not.

(b) Consider the GMRES algorithm applied to an n by n matrix A with the sparsity pattern pictured below:

$$\begin{bmatrix} * & * & \cdots & * & * \\ * & * & \cdots & * & 0 \\ 0 & * & \cdots & * & 0 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \cdots & \cdots & * & 0 \end{bmatrix},$$

where the $*$'s represent nonzero entries. Show that if the initial residual is the nth unit vector $(0, \ldots, 0, 1)^T$, then the algorithm makes no progress until step n. Show that a matrix with this sparsity pattern can have *any* eigenvalues. Conclude that eigenvalue information alone cannot be enough to ensure fast convergence of the GMRES algorithm.

Solution

1. Recall $\|Ax\|_2 \leq \|A\|_2 \|x\|_2$.

   We therefore have,

   $$\|e_k\|_2 = \left\|(I - M^{-1}A)^k e_0\right\| \leq \left\|I - M^{-1}A\right\|^k \|e_0\| = 2^{-k}$$

   Therefore we have $\|e_k\| < 2^{-20}$ when $k > 20$.

   This is an upper bound, so depending on M and A the bound could be improved.

   We have $\rho(A) \leq \|A\|_2$. However there is no relation in the other direction for general matrices A.

   Without knowing the structure of M we cannot bound the number of iterations required to reach some tolerance.

   If we knew M we might be able to find a bound which does not depend on A for some set of reasonable A (although this is not true for all A for the same reasons as above).

   Otherwise, if we know $I - M^{-1}A$ is symmetric, then $\rho(I - M^{-1}A) = \left\|I - M^{-1}A\right\|_2$ so we can use the above estimate.

2. Observe that $Ar_0 = (a_{1,n}, 0, \ldots, 0)^T$. Therefore $A^2 r_0$ has sparsity pattern $(*, *, 0 \ldots, 0)^T$ and $A^3 r_0$ has sparsity pattern $(*, *, *, 0, \ldots, 0)^T$. Therefore, r_0 is orthogonal to $A^k r_0$ for $k = 1, \ldots, n-1$.

At step k the residual has the form,

$$r_k = r_0 - AQ_k y_k$$

Recall the columns of Q_k form a basis for $\mathcal{K}_k$. Therefore,

$$AQ_k y_k \in \operatorname{span}\{Ar_0, A^2 r_0, \ldots, A^k r_0\}$$

Since $AQ_k y_k \in \operatorname{span}\{Ar_0, A^2 r_0, \ldots, A^k r_0\}$, then for any $k < n$ we know r_k has a one in the last slot. Therefore $\|r_k\|_2 \geq 1$. However since AQ_k is not full rank for $k < n$, clearly there is some y_k such that $AQ_k y_k = 0$ so that $\|r_k\|_2 = 1$.

Since the algorithm minimizes the residual it will pick $y_k \in \ker AQ_k$ at each of these steps.

This means the algorithm makes no progress for the first $n-1$ steps. However, at step n the algorithm is able to make progress.

Let $\{\lambda_k\}_{k=1}^n$ be any n complex numbers. Then there is a polynomial $p(x) = c_0 + c_1 x + \ldots + c_{n-1} x^{n-1} + t^n$ over $\mathbb{C}$ with each $\overline{\lambda}_k$ as a root.

Let C be the companion matrix of $p(x)$. Then each $\overline{\lambda}_k$ is an eigenvalue of C.

Let Q be the matrix with ones on the antidiagonal. Then Q is unitary so that $Q^*Q = I$.

Explicitly we have,

$$C = \begin{bmatrix} 0 & 0 & \cdots & 0 & -c_0 \\ 1 & 0 & \cdots & 0 & -c_1 \\ 0 & 1 & \cdots & 0 & -c_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -c_{n-1} \end{bmatrix} \qquad Q = \begin{bmatrix} & & & & 1 \\ & & & 1 & \\ & & \iddots & & \\ & 1 & & & \\ 1 & & & & \end{bmatrix}$$

Observe,

$$(QCQ^*)^* = \begin{bmatrix} -c_{n-1} & \cdots & -c_2 & -c_1 & -c_0 \\ 1 & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \cdots & 1 & 0 & 0 \\ 0 & \cdots & 0 & 1 & 0 \end{bmatrix}$$

Note that QCQ^* is a similarity transform of C and therefore the eigenvalues of QCQ^* are the λ_k. The eigenvalues of the Hermetian conjugate of a matrix are the complex conjugate of the eigenvalues of the original matrix. This proves λ_k is an eigenvalue of $(Q^*CQ)^*$ for $k = 1, 2, \ldots, n$.

Clearly this matrix has the desired form (in fact even stricter). Therefore, a (real) matrix with this sparsity pattern can have any (real or complex) eigenvalue.

Together these two results mean that given any eigenvalues there is some matrix with these eigenvalues for with which GMRES does not make progress on until the n-th step.