Optimización numérica

Proyecto 1. Precondicionamiento

Instructor José Luis Morales. Otoño, 2015

Gradiente conjugado precondicionado

El método de gradiente conjugado (GC) pertenece a la familia de algoritmos basados en subespacios de Krylov; el método está diseñado para resolver sistemas de ecuaciones lineales de la forma

$$Ax = b, \tag{1}$$

en donde A es una matriz simétrica positiva definida de $n \times n$. En aritmética exacta GC termina en no más de n iteraciones. Sin embargo, en aritmética de punto flotante, GC es considerado un método iterativo.

Las aplicaciones en las que el GC es particularmente versátil y efectivo son aquellas en las que n está en el orden de las decenas o cientos de miles y A es rala (típicamente con un número de entradas diferentes de cero de orden lineal en n); en estos casos la solución se alcanza en un número de iteraciones sustancialmente menor que n.

Por otra parte, se sabe que la eficiencia del método es sensible a la distribución de valores propios de A; en términos prácticos el número de iteraciones es aproximadamente igual al número de cúmulos de valores propios de A. Una estrategia muy popular para disminuir el número de cúmulos es el uso de precondicionadores.

El principio mediante el cual funciona un precondicionador es relativamente simple. Si el sistema por resolver es (1), un precondicionador es una matriz M simétrica positiva definida cuya inversa aproxima a la inversa de A. Por lo tanto, se espera que la matriz del sistema *precondicionado*

$$M^{-1}Ax = M^{-1}b \tag{2}$$

tenga una distribución de valores propios más favorable que la de A y que GC aplicado al sistema anterior termine en menos iteraciones que las que requiere para resolver el sistema original (1). También se espera que el costo computacional compense en mucho la inversión en al elección y construcción de M.

En el proyecto vamos a utilizar una forma ligeramente diferente para construir un precondicionador. Supondremos que $M = C^T C$, con C invertible, por lo tanto M es simétrica positiva definida. Las siguientes definiciones

$$\bar{x} = Cx, \quad \bar{A} = C^{-T}AC^{-1}, \quad \bar{b} = C^{-T}b$$

establecen el sistema precondicionado

$$\bar{A}\bar{x} = \bar{b}.$$

al cual aplicaremos el método de GC. El algoritmo resultante es

Gradiente conjugado precondicionado (GCP)

Escoger x_0, una aproximación inicial; $\quad r_0 \leftarrow Ax_0 - b$,

$y_0 \leftarrow M^{-1}r_0, \quad p_0 \leftarrow -y_0, \quad k \leftarrow 0.$

Repetir mientras $||r_k|| > TOL$

$$\begin{aligned}
\alpha_k &= \frac{r_k^T y_k}{d_k^T A d_k} \\
x_{k+1} &= x_k + \alpha_k p_k \\
r_{k+1} &= r_k + \alpha_k A p_k \\
y_{k+1} &= M^{-1} r_{k+1} \\
\beta_{k+1} &= \frac{r_{k+1}^T y_{k+1}}{r_k^T y_k} \\
p_{k+1} &= -y_{k+1} + \beta_{k+1} p_k \\
k &\leftarrow k+1
\end{aligned}$$

Las factorizaciones incompletas de A son probablemente los precondicionadores más populares y versátiles en diversas aplicaciones. En el proyecto vamos a utilizar factorizaciones incompletas de Cholesky de A, es decir que $M = LL^T$. Observar que en el algoritmo anterior el cálculo $y_k = M^{-1}r_k$ equivale a resolver el sistema

$$LL^T y_k = r_k \tag{3}$$

una vez por cada iteración; al vector y_k se le conoce como el *residuo precondicionado*. El costo por iteración de GCP, con respecto a GC, se incrementa en un vector de memoria para almacenar y_k y el requerido para resolver el sistema (3).

Proyecto

El objetivo del proyecto es verificar experimentalmente las propiedades computacionales de GCP. El ambiente computacional es MATLAB; con ciertas limitaciones, el proyecto se puede realizar en otros ambientes como OCTAVE, SCILAB, PYTHON, ...

Actividades

1. Escribir una versión computacional de GC. Fijar la condición de paro como

$$||r_k||_2 \leq ||r_0||_2\, TOL, \quad TOL = 1 \times 10^{-8}.$$

2. Comparar la versión de GC con el comando `pcg`. Utilizar matrices simétricas positivas definidas de la colección `gallery`, por ejemplo `lehmer, moler, poisson, toeppd`.

3. Modificar GC para aceptar precondicionamiento. Los precondicionadores se pueden obtener llamando al comando `ichol`. El comando `ichol` construye diferentes variantes de factorizaciones incompletas de Cholesky. Utilizar la versión más simple `L = ichol(A)` y la modificación conocida como *modified incomplete Cholesky*, accesible con la opción `michol = 'on'`.

4. Comparar el tiempo de CPU requerido para resolver $Ax = b$ mediante

   GC GCP `pcg` `chol`

   En donde `chol` se refiere a la factorización de Cholesky. Es decir que compararemos el desempeño de un método directo con diversas variantes del método de gradiente conjugado precondicionado.

   Utilizar matrices ralas `poisson` y `toeppd` de dimensión variable. Fijar la tolerancia para la condición de paro como $TOL = 1 \times 10^{-8}$. Utilizar $x_0 = [0, \ldots, 0]^T$ como punto inicial. Generar los lados derechos b mediante el siguiente comando `b = A*ones(n,1)`.

5. Escribir el reporte técnico.