Scientific Computing II

Relaxation Methods

Michael Bader

Summer term 2012

The Residual Equation

- for $Ax = b$, we define the **residual**:

$$r^{(i)} = b - Ax^{(i)}$$

- and the error: $e^{(i)} := x - x^{(i)}$
  (thus $x := x^{(i)} + e^{(i)}$);
- short computation:

$$r^{(i)} = b - Ax^{(i)} = Ax - Ax^{(i)} = A(x - x^{(i)}) = Ae^{(i)}.$$

- **residual equation:**

$$Ae^{(i)} = r^{(i)}$$

Residual Based Correction

Solve $Ax = b$ using the residual equation $Ae^{(i)} = r^{(i)}$

- r (which can be computed) is an indicator for the size of the error e (which is not known).
- use residual equation to compute a *correction* to $x^{(i)}$
- basic idea: solve a modified (easier) SLE:

$$B\,\hat{e}^{(i)} = r^{(i)} \qquad \text{where} \quad B \sim A$$

- use $\hat{e}^{(i)}$ as an approximation for $e^{(i)}$, and set

$$x^{(i+1)} = x^{(i)} + \hat{e}^{(i)}.$$

Relaxation

How should we choose B?

- $B \sim A$ (B "similar" to A),
  i.e. $B^{-1} \approx A^{-1}$,
  or at least $B^{-1}y \approx A^{-1}y$ for most vectors y.
- $Be = r$ should be easy/fast to solve

Examples:

- $B = \text{diag}(A) = D_A$ (diagonal part of A)
  $\Rightarrow$ Jacobi iteration
- $B = L_A$ (lower triangular part of A)
  $\Rightarrow$ Gauss-Seidel iteration

Jacobi Relaxation

Iteration formulas in matrix-vector notation:

1. residual notation:

$$x^{(i+1)} = x^{(i)} + D_A^{-1} r^{(i)} = x^{(i)} + D_A^{-1} \left(b - A x^{(i)} \right)$$

2. for implementation:

$$x^{(i+1)} = D_A^{-1} \left(b - (A - D_A) x^{(i)} \right)$$

3. for analysis:

$$x^{(i+1)} = \left(I - D_A^{-1} A \right) x^{(i)} + D_A^{-1} b =: M x^{(i)} + N b$$

Jacobi Relaxation – Algorithm

- based on: $x^{(i+1)} = D_A^{-1}\left(b - (A - D_A)x^{(i)}\right)$

```
for i from 1 to n do
    xnew[i] := ( b[i]
          - sum( A[i,j]*x[j], j=1..i-1)
          - sum( A[i,j]*x[j], j=i+1..n)
        ) / A[i,i];
end do;
for i from 1 to n do
    x[i] := xnew[i];
end do;
```

- **properties:**
  - additional storage required (`xnew`)
  - `x`, `xnew` can be computed **in any order**
  - `x`, `xnew` can be computed **in parallel**

Gauss-Seidel Relaxation

Iteration formulas in matrix-vector notation:

1. residual notation:

$$x^{(i+1)} = x^{(i)} + L_A^{-1} r^{(i)} = x^{(i)} + L_A^{-1} \left(b - A x^{(i)} \right)$$

2. for implementation:

$$x^{(i+1)} = L_A^{-1} \left(b - (A - L_A) x^{(i)} \right)$$

3. for analysis:

$$x^{(i+1)} = \left(I - L_A^{-1} A \right) x^{(i)} + L_A^{-1} b =: M x^{(i)} + N b$$

Gauss-Seidel Relaxation – Algorithm

- based on: $x^{(i+1)} = L_A^{-1}\left(b - (A - L_A)x^{(i)}\right)$
- solve $L_A x^{(i+1)} = b - (A - L_A)x^{(i)}$
  via backwards substitution:

```
for i from 1 to n do
    x[i] := ( b[i]
          - sum( A[i,j]*x[j], j=1..i-1)
          - sum( A[i,j]*x[j], j=i+1..n)
        ) / A[i,i];
end do;
```

- **properties:**
  - no additional storage required
  - inherently **sequential** computation of `x`
  - usually faster than Jacobi

Successive-Over-Relaxation (SOR)

- observation: Gauss-Seidel corrections are "too small"
- add an over-relaxation-factor α:

```
for i from 1 to n do
    x[i] := x[i] + alpha * ( b[i]
          - sum( A[i,j]*x[j], j=1..n)
        ) / A[i,i];
end do;
```

- for 2D-Poisson: optimal α (≈ 1.7) improves convergence: $\mathcal{O}(n^2) \rightarrow \mathcal{O}(n^{3/2})$

Does It Always Work?

- simple answer: no (life is not that easy . . .)
- Jacobi: matrix A needs to be *diagonally dominant*
- Gauß-Seidel: matrix A needs to be *positive definite*
- How about performance?
  $\rightarrow$ usually quite slow

Our next topics:

1. How slow are the methods exactly?
2. What is the underlying reason?
3. Is there a fix?