Forward Euler Method

Forward Euler is an "explicit method"

$$\mathbf{Q}^{n+1} = \mathbf{Q}^n + \Delta t A \mathbf{Q}^n$$
$$= (I + \Delta t A)\mathbf{Q}^n$$

In order to ensure that $\| \mathbf{Q}^{n+1} \|$ doesn't grow, we needed

$$\| \mathbf{Q}^{n+1} \| = \| (I + \Delta t A)\mathbf{Q}^n \| < \| I + \Delta t A \| \| \mathbf{Q}^n \|$$

or $\| I + \Delta t A \| < 1$. We chose our time step so that

$$-1 < 1 + \Delta t \lambda^p(A) < 1$$

or that

$$(\Delta t)_{stable} < \frac{(\Delta x)^2}{2}$$

time step restriction

Backward Euler Method

Backward Euler is an "implicit method"

$$\begin{aligned}\mathbf{Q}^{n+1} &= \mathbf{Q}^n + \Delta t A \mathbf{Q}^{n+1} \\ &= (I - \Delta t A)^{-1} \mathbf{Q}^n\end{aligned}$$

Just as with FE, we need to ensure that $\| \mathbf{Q}^{n+1} \|$ doesn't grow.

$$\| \mathbf{Q}^{n+1} \| = \| (I - \Delta t A)^{-1} \mathbf{Q}^n \| < \frac{\| \mathbf{Q}^n \|}{\| I - \Delta t A \|}$$

We have

$$0 < \frac{1}{1 - \Delta t \lambda^p(A)} < 1$$

no time step restriction

for all $\Delta t > 0$ and the Backward Euler method is *unconditionally stable*. Typically, however, we try to take $\Delta t \approx \Delta x$

The cost of an implicit method

Implicit methods are have better stability properties than explicit methods, but each time step is more costly, and can be more difficult to implement (especially in parallel).

Explicit (Forward Euler)

```
Initialize Q(0)
for k = 0,1,2, ....
    Q(n+1) = Q(n) + dt*A*Q(n);
```

"embarassingly parallel" and requires only local communication

Implicit (Backward Euler)

```
Initialize Q(0)
for k = 0,1,2, ....
   Solve (I - dt*A)Qnp1 = Qn;
   Q(n+1) = Qnp1;
```

non-trivial to parallelize and requires global communication.

Solving linear systems

At each time step of Backward Euler, we have to solve the linear system

$$(I - \Delta t A)\mathbf{x} = \mathbf{Q}^n$$

for the unknowns $\mathbf{x}$.

In what follows, we assume that we are solving the "canonical" linear system given by

$$A\mathbf{x} = \mathbf{b}$$

For the heat equation using backward Euler, we have

$$A \leftarrow I - \Delta t A$$

$$\mathbf{b} \leftarrow \mathbf{Q}^n$$

$$\mathbf{x} \leftarrow \mathbf{Q}^{n+1}$$

Splitting methods

Splitting Methods use a “fixed point iteration” and solve

$$g(\mathbf{x}) = \mathbf{x} + M^{-1}(\mathbf{b} - A\mathbf{x}) = x$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + M^{-1}(\mathbf{b} - A\mathbf{x})$$

The Jacobi Method chooses $M = D$, the diagonal of A.

The Gauss-Seidel method choose $M = L + D$, the lower triangular portion of A.

Convergence properties depend on characteristics of the matrix

SPD = symmetric positive definite is typical

Steepest-descent

Steepest-descent finds the minimum of the function

$$F(\mathbf{x}) = \mathbf{x}^T A\mathbf{x} - 2\mathbf{b}^T\mathbf{x}$$

Scalar function

The direction of greatest *increase* of this function is the gradient

$$\nabla F(\mathbf{x}) = 2(A\mathbf{x} - \mathbf{b})$$

and so the "descent direction" is the negative gradient. Iterates take the form

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{x}_k - a_k \nabla F(\mathbf{x}_k) \\ &= \mathbf{x}_k + a_k \mathbf{r}_k \qquad \text{(ignore factor of 2)}\end{aligned}$$

where

$$\mathbf{r}_k = \mathbf{b} - A\mathbf{x}_k$$

Residual

and a_k is chosen to minimize the error in the search direction.

Conjugate Gradient

The steepest descent algorithm is an enormous improvement over Jacobi or Gauss-Seidel. But it can also be improved upon by choosing better search directions.

Given an initial guess x_0, compute r_0 and set $p_0 = r_0$.
For k = 0,1,2,3,...

- Compute Ap_k
- Set $x_{k+1} = x_k + a_k p_k$ where a_k is chosen to minimize the error in the A-norm.
- Compute $r_{k+1} = r_k - a_k A p_k$
- Set $p_{k+1} = r_k + b_k p_k$, where b_k are chose to minimize the residual in the A-norm.

Implementing iterative methods

We need :

- A “matrix vector multiply” that is easy to implement
- A dot product operator that is easy to implement.
- Vector operations of the form

$$\mathbf{z} = \mathbf{x} + a\mathbf{y}$$

These are all straightforward to implement in parallel.