Scientific Computing

A practical Companion

6th Notebook

This notebook can be downloaded from the location:
http : //
staff.science.uva.nl / ~walter / SC / Notebooks / SC08 – 6. nb

Author:

Walter Hoffmann (Korteweg-de Vries Institute for Mathematics, UvA)

January - March, 2007

Systems of linear equations III
Iterative solution methods by

fixed-point iteration, part 1

General stationary iterative method

Let a system of linear equations be:

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

and assume that A is invertible so that the system has a unique solution which may be written as $\boldsymbol{x} = \boldsymbol{A}^{-1}\,\boldsymbol{b}$.

If the system is not very large, the simplest way of calculating its solution is by so called *direct* methods (as opposed to *iterative* methods).

An iterative method is called *stationary* if the matrix that is used during the iterative process of approximating the solution, does not change during these iterations; "*no information is fed back into the process*".

The basic idea is that a matrix F (say) is found, which is in some sense an approximation to A and for which solving a system is relatively simple (easy and cheap).

If matrix F is given then we may write:

$$(\boldsymbol{A} - \boldsymbol{F})\,\boldsymbol{x} + \boldsymbol{F}\,\boldsymbol{x} = \boldsymbol{b},$$

or:

$$\begin{aligned}\boldsymbol{F}\,\boldsymbol{x} &= \boldsymbol{b} - (\boldsymbol{A} - \boldsymbol{F})\,\boldsymbol{x}\,.\\ &= \boldsymbol{b} + (\boldsymbol{F} - \boldsymbol{A})\,\boldsymbol{x}\end{aligned}$$

In the form of a *stationary* or *fixed-point* iteration, this looks like

$$\boldsymbol{F}\,\boldsymbol{x}^{(i+1)} = \boldsymbol{b} + (\boldsymbol{F} - \boldsymbol{A})\,\boldsymbol{x}^{(i)}$$

giving:

$$\boldsymbol{x}^{(i+1)} = \boldsymbol{F}^{-1}\,\boldsymbol{b} + \left(\boldsymbol{I} - \boldsymbol{F}^{-1}\,\boldsymbol{A}\right)\boldsymbol{x}^{(i)}$$

for which the stationary value (a fixed point) is clearly the solution $\overline{\boldsymbol{x}}$ of the original system $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$.

We ask ourselves: "under what conditions is this iteration convergent?"

Define $\boldsymbol{e}^{(i)} = \overline{\boldsymbol{x}} - \boldsymbol{x}^{(i)}$; the so called *error* in the i-th iterate (which is of course unknown!). Then we have:

$$\begin{aligned}\boldsymbol{e}^{(i+1)} &= \overline{\boldsymbol{x}} - \boldsymbol{x}^{(i+1)} = \boldsymbol{A}^{-1}\,\boldsymbol{b} - \boldsymbol{x}^{(i+1)}\\ &= \boldsymbol{A}^{-1}\,\boldsymbol{b} - \boldsymbol{F}^{-1}\,\boldsymbol{b} - \left(\boldsymbol{I} - \boldsymbol{F}^{-1}\,\boldsymbol{A}\right)\boldsymbol{x}^{(i)}\\ &= \boldsymbol{A}^{-1}\,\boldsymbol{b} - \boldsymbol{x}^{(i)} - \boldsymbol{F}^{-1}\,\boldsymbol{b} + \boldsymbol{F}^{-1}\,\boldsymbol{A}\,\boldsymbol{x}^{(i)}\end{aligned}$$

$$= e^{(i)} - F^{-1}\left(A A^{-1} b - A x^{(i)}\right)$$
$$= e^{(i)} - F^{-1} A\left(A^{-1} b - x^{(i)}\right)$$
$$= \left(I - F^{-1} A\right) e^{(i)}.$$

This can be expressed in terms of the initial error by recursively applying this formula:

$$e^{(i+1)} = \left(I - F^{-1} A\right) e^{(i)}$$
$$= \left(I - F^{-1} A\right)^2 e^{(i-1)}$$
$$\cdots$$
$$= \left(I - F^{-1} A\right)^{i+1} e^{(0)}$$

Using norms, we have:

$$\| e^{(i+1)} \| \leq \| \left(I - F^{-1} A\right)^{i+1} \| \, \| e^{(0)} \|$$
$$\leq \| \left(I - F^{-1} A\right) \| \, \| \left(I - F^{-1} A\right)^{i} \| \, \| e^{(0)} \|$$
$$\cdots$$

yielding:

$$\| e^{(i+1)} \| \leq \| I - F^{-1} A \|^{i+1} \, \| e^{(0)} \| .$$

From this we conclude that the iteration does converge whenever a norm of the matrix $\left(I - F^{-1} A\right)$ is less than 1 .

Instead of looking at the error $e^{(i)}$ we could also look at the size of the residual vector $r^{(i)}$ defined by $r^{(i)} = b - A x^{(i)}$.

First we formulate the relation between $e^{(i)}$ and $r^{(i)}$ (the so called *residual equation*):

$$A e^{(i)} = A\left(\overline{x} - x^{(i)}\right)$$
$$= b - A x^{(i)} = r^{(i)}.$$

Now we remember the relation between $e^{(i+1)}$ and $e^{(i)}$:

$$e^{(i+1)} = \left(I - F^{-1} A\right) e^{(i)}$$

which we may write as:

$$A e^{(i+1)} = A\left(I - F^{-1} A\right) e^{(i)}$$

giving:

$$\begin{aligned} r^{(i+1)} &= \left(A - A F^{-1} A\right) e^{(i)} \\ &= \left(I - A F^{-1}\right) A\, e^{(i)} \\ &= \left(I - A F^{-1}\right) r^{(i)}. \end{aligned}$$

From this we conclude that also the inequality

$$\| I - A F^{-1} \| < 1$$

assures convergence.

Defect correction formulation

We have seen the fixed-point iteration form for an iterative method for solving systems of linear equations:

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

or alternatively:

$$F x^{(i+1)} = b + (F - A) x^{(i)}.$$

If seen as the starting point for an algorithm to iteratively solve a linear system, this can be formulated with explicit use of the residual vector $r^{(i)}$ $(= b - A x^{(i)})$ as follows

$$F x^{(i+1)} = r^{(i)} + F x^{(i)}$$

giving:

$$F x^{(i+1)} - F x^{(i)} = r^{(i)}$$

which can be written as:

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

This suggests the following algorithm:

Take some initial guess $x^{(0)}$ (it needs not to be a good approximation; the zero-vector is O.K.)

For $i = 0, 1, 2,$ until convergence DO :

Calculate:	$r^{(i)} = b - A x^{(i)}$	("which is the defect")
Solve for $d^{(i)}$:	$F d^{(i)} = r^{(i)}$	("correct")
Add:	$x^{(i+1)} = x^{(i)} + d^{(i)}$	("adjust")

Various choices for F lead to various methods .

If the convergence condition 'barely' holds; that is to say, if none of the quantities $\| I - F^{-1} A \|$ or $\| I - A F^{-1} \|$ is (very) small, then convergence may be very slow.

Even if the correctionvector $\boldsymbol{d}^{(i)}$ is in some sense small (two consecutive iterands $x^{(i)}$ and $x^{(i+1)}$ are close together), then the current iterate may still be far away from the limiting (stationary) value.

This is stated in the following

Theorem

$$\text{If } \| I - F^{-1} A \| = \delta, \text{ then } \| x^{(i+1)} - \bar{x} \| \leq \frac{\delta}{1-\delta} \| x^{(i+1)} - x^{(i)} \|$$

Proof:

$$e^{(i+1)} = (I - F^{-1} A)\, e^{(i)}$$
$$\| x^{(i+1)} - \bar{x} \| \leq \| I - F^{-1} A \| \, \| x^{(i)} - \bar{x} \|$$
$$\| x^{(i+1)} - \bar{x} \| \leq \delta \, \| x^{(i)} - x^{(i+1)} + x^{(i+1)} - \bar{x} \|$$
$$\leq \delta \, (\| x^{(i)} - x^{(i+1)} \| + \| x^{(i+1)} - \bar{x} \|)$$

giving:

$$\| x^{(i+1)} - \bar{x} \| \, (1 - \delta) \leq \delta \, \| x^{(i)} - x^{(i+1)} \|$$

from which follows:

$$\| x^{(i+1)} - \bar{x} \| \leq \frac{\delta}{1-\delta} \| x^{(i+1)} - x^{(i)} \| .$$

We investigate various methods via various choices for the so called

splitting matrix F.

Jacobi, Gauss-Seidel and SOR method

One of the most simplest iterative methods for solving systems of linear equations is widely known as Jacobi's [1] iterative method. The method was known to, and has been used by the famous mathematician C.F. Gauss [2] and we think that he was the first one to use this method. Therefore it should be called: Gauss-Jacobi's method.

This method is based on a choice for the splitting matrix F, given by $F = \text{diag}(A)$, where $\text{diag}(A)$ is the diagonal matrix consisting of the diagonal elements of A.

If A is a *diagonal dominant* matrix (each diagonal element $a_{i,i}$ is in absolute value larger than the sum of the absolute values of the elements in that row: $|a|_{i,i} \geq \sum_{j=1;j\neq i}^{n} |a_{i,j}|$) then from Gerschgorin's theorem we may conclude that all eigenvalues of the matrix $D^{-1} A$ are bounded by 1 and that therefore Gauss-Jacobi is convergent.

However, it may be very slowly converging as is the case for matrices $D^{-1} A$ that have a spectral radius which is barely less than 1 .

If we use as a splitting matrix either the lower triangular or upper triangular part of the matrix, we have the variant that is known by the name of Gauss-Seidel [3] iteration. A more sofisticated variant of Gauss-Seidel iteration is the method of Successive Over Relaxation or for short SOR method. Both Gauss-Seidel and SOR are treated in the following examples.

[1]) Carl Gustav Jacob Jacobi 1804 - 1851

[2]) Johann Carl Friedrich Gauss 1777 - 1855

[3]) Philipp von Seidel 1821 - 1896

▽ Implementations:

Let A be the matrix that arises from a discretization as in the assignment for the steady state diffusion equation with initial and boundary values as given. We take 5 subintervals, both in x- and y-direction, defining 6 gridlines (including $x = 0$, $x = 1$ and $y = 0$, $y = 1$). Because of the identification of left and right boundary, we have 5 lines of unknowns in x-direction and 4 lines of unknowns in y-direction defining 20 equations for 20 unknowns. For reasons to become clear later, we work with the opposite of the matrix and therefore also with the opposite of the right-hand side vector. So all the diagonal elements of A are positive; A itself is positive definite.

▽ Matrix A:

In[144]:=

```
A = {{4, -1, 0, 0, -1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
     {-1, 4, -1, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
     {0, -1, 4, -1, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
     {0, 0, -1, 4, -1, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
     {-1, 0, 0, -1, 4, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
     {-1, 0, 0, 0, 0, 4, -1, 0, 0, -1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0},
     {0, -1, 0, 0, 0, -1, 4, -1, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0},
     {0, 0, -1, 0, 0, 0, -1, 4, -1, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0},
     {0, 0, 0, -1, 0, 0, 0, -1, 4, -1, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0},
     {0, 0, 0, 0, -1, -1, 0, 0, -1, 4, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0},
     {0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 4, -1, 0, 0, -1, -1, 0, 0, 0, 0},
     {0, 0, 0, 0, 0, 0, -1, 0, 0, 0, -1, 4, -1, 0, 0, 0, -1, 0, 0, 0},
     {0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, -1, 4, -1, 0, 0, 0, -1, 0, 0},
     {0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, -1, 4, -1, 0, 0, 0, -1, 0},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, -1, 4, 0, 0, 0, 0, -1},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 4, -1, 0, 0, -1},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, -1, 4, -1, 0, 0},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, -1, 4, -1, 0},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, -1, 4, -1},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, -1, 0, 0, -1, 4}};
```

▽ Right-hand side:

In[145]:=

```
b = {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 1., 1., 1., 1., 1.};
```

▽ Check on the solution by direct method:

In[146]:=

```
u = LinearSolve[A, b]
```

Out[146]=

```
{0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.4, 0.4, 0.4, 0.4,
 0.6, 0.6, 0.6, 0.6, 0.6, 0.8, 0.8, 0.8, 0.8, 0.8}
```

▽ Diagonal of A:

In[147]:=

```
DD = DiagonalMatrix[
       {4, 4, 4, 4, 4, 4, 4, 4, 4, 4,
         4, 4, 4, 4, 4, 4, 4, 4, 4, 4}];
```

In[148]:=

```
DD // MatrixForm
```

Out[148]//MatrixForm=

$$\begin{pmatrix}
4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4
\end{pmatrix}$$

▽ Jacobi's iterative algorithm:

or rather Gauss-Jacobi:

▽ (On the speed of convergence)

How many steps are expected to diminish the error by a factor of 10?

In[149]:=

```
Id = IdentityMatrix[20];
```

In[150]:=

```
q = N[Norm[ Id  – A.Inverse[DD]]]
```

Out[150]=

```
0.904508
```

In[151]:=

```
f = Log[0.1] / Log[q]
```

Out[151]=

```
22.9424
```

In[152]:=

```
x = {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.};

i = 0;
```

In[264]:=

```
r = b  –  A.x;
r.r
i = i + 1;
d = LinearSolve[DD, r];
x = x +  d;
```

Out[265]=

```
0.00834881
```

In[269]:=

```
i
x
```

Out[269]=

```
23
```

Out[270]=

```
{0.164043, 0.164043, 0.164043, 0.164043, 0.164043,
 0.341806, 0.341806, 0.341806, 0.341806, 0.341806,
 0.541787, 0.541787, 0.541787, 0.541787, 0.541787,
 0.764013, 0.764013, 0.764013, 0.764013, 0.764013}
```

In[271]:=

```
er = u – x;
er.er
```

Out[272]=

```
0.0468166
```

▽ Strictly upper-triangular part of A:

In[273]:=

```
U = {{0, –1, 0, 0, –1, –1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
     {0, 0, –1, 0, 0, 0, –1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
     {0, 0, 0, –1, 0, 0, 0, –1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
     {0, 0, 0, 0, –1, 0, 0, 0, –1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, –1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
     {0, 0, 0, 0, 0, 0, –1, 0, 0, –1, –1, 0, 0, 0, 0, 0, 0, 0, 0, 0},
     {0, 0, 0, 0, 0, 0, 0, –1, 0, 0, 0, –1, 0, 0, 0, 0, 0, 0, 0, 0},
     {0, 0, 0, 0, 0, 0, 0, 0, –1, 0, 0, 0, –1, 0, 0, 0, 0, 0, 0, 0},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, –1, 0, 0, 0, –1, 0, 0, 0, 0, 0, 0},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, –1, 0, 0, 0, 0, 0},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, –1, 0, 0, –1, –1, 0, 0, 0, 0, 0},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, –1, 0, 0, 0, –1, 0, 0, 0, 0},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, –1, 0, 0, 0, –1, 0, 0, 0},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, –1, 0, 0, 0, –1, 0, 0},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, –1, 0, 0, 0, –1, 0},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, –1, 0, 0, 0, –1},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, –1, 0, 0, –1},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, –1, 0, 0},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, –1, 0},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, –1},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}};
```

▽ Strictly lower-triangular part of A:

In[274]:=

```
L = {{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
     {-1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
     {0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
     {0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
     {-1, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
     {-1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
     {0, -1, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
     {0, 0, -1, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
     {0, 0, 0, -1, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
     {0, 0, 0, 0, -1, -1, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
     {0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},
     {0, 0, 0, 0, 0, 0, -1, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0},
     {0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0},
     {0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, -1, -1, 0, 0, -1, 0, 0, 0, 0, 0, 0},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, -1, 0, 0, 0, 0},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, -1, 0, 0, 0},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, 0, 0, 0, -1, 0, 0},
     {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1, -1, 0, 0, -1, 0}};
```

▽ Gauss-Seidel's iterative algorithm:

What is the difference between using $L + D$ and $U + D$?

We show both; we start with the $U + D$ variant (= backward relaxation):

In[275]:=

```
x = {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.};

i = 0;
```

In[332]:=

```
r = b − A.x;
r.r
i = i + 1;
d = LinearSolve[U + DD, r];
x = x + d;
```

Out[333]=

```
0.00889246
```

In[337]:=

```
i
x
```

Out[337]=

```
12
```

Out[338]=

```
{0.176241, 0.175279, 0.174277, 0.173234, 0.17215,
 0.357537, 0.355816, 0.354025, 0.352161, 0.350221,
 0.55309, 0.551188, 0.549207, 0.547146, 0.545,
 0.767969, 0.766668, 0.765314, 0.763904, 0.762437}
```

In[339]:=

```
er = u − x;
er.er
```

Out[340]=

```
0.0330004
```

and now for the $L + D$ variant (= forward relaxation):

In[341]:=

```
x = {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.};

i = 0;
```

In[408]:=
```
r = b − A.x;
r.r
i = i + 1;
d = LinearSolve[L + DD, r];
x = x + d;
```

Out[409]=
```
0.00727135
```

In[413]:=
```
i
x
```

Out[413]=
```
14
```

Out[414]=
```
{0.166033, 0.167356, 0.168627, 0.169848, 0.171021,
 0.350221, 0.352161, 0.354025, 0.355816, 0.357537,
 0.554919, 0.556677, 0.558366, 0.559989, 0.561548,
 0.774772, 0.775756, 0.776701, 0.777609, 0.778482}
```

In[415]:=
```
er = u − x;
er.er
```

Out[416]=
```
0.0270416
```

SOR algorithm:

In both variants of Gauss-Seidel's algorithm, it appears that given the direction of the correctionvector d, its size is suboptimal. In all iterands it appears that a step in the direction of d, but with a somewhat larger size yields a better convergent process. The factor that is used for stretching d may not be chosen to larger than 2, as can be proven. The optimal value for this *over relaxation factor,* ω, is problem dependent. A good choice in general for a wide range of problems appears to be somewhere around 1.5 or 1.6

Again we start with the $U + D$ variant (= backward relaxation):

In[417]:=
```
x = {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.};

i = 0;
```

In[459]:=

```
r = b − A.x;
r.r
i = i + 1;
d = LinearSolve[U + DD, r];
x = x + 1.6 d;
```

Out[460]=

```
0.00468027
```

In[464]:=

```
i
x
```

Out[464]=

```
9
```

Out[465]=

```
{0.188063, 0.187247, 0.187123, 0.186069, 0.186516,
 0.37849, 0.377499, 0.376934, 0.374952, 0.375643,
 0.576454, 0.574596, 0.57565, 0.570887, 0.573879,
 0.784362, 0.78094, 0.785456, 0.77905, 0.77896}
```

In[466]:=

```
er = u – x;
er.er
```

Out[467]=

```
0.00859196
```

And now for the "L + D" variant.

In[468]:=

```
x = {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.};

i = 0;
```

In[580]:=

```
r = b − A.x;
r.r
i = i + 1;
d = LinearSolve[L + DD, r];
x = x + 1.6 d;
```

Out[581]=

```
0.00675166
```

In[585]:=

```
i
x
```

Out[585]=

```
23
```

Out[586]=

```
{0.19023, 0.205708, 0.196112, 0.202319, 0.198133,
 0.399558, 0.399893, 0.399356, 0.400009, 0.399537,
 0.599203, 0.600094, 0.599507, 0.599868, 0.599672,
 0.799631, 0.799984, 0.799752, 0.799927, 0.799809}
```

In[587]:=

```
er = u – x;
er.er
```

Out[588]=

```
0.000154103
```

Try for a smaller value than 1.6:

In[589]:=

```
x = {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.};

i = 0;
```

In[651]:=
```
r = b − A.x;
r.r
i = i + 1;
d = LinearSolve[L + DD, r];
x = x + 1.5 d;
```

Out[652]=
```
0.00857443
```

In[656]:=
```
i
x
```

Out[656]=
```
13
```

Out[657]=
```
{0.183404, 0.198101, 0.187821, 0.19417, 0.190317,
 0.388394, 0.384829, 0.388984, 0.387629, 0.38884,
 0.587278, 0.58918, 0.588291, 0.589539, 0.589549,
 0.793192, 0.793457, 0.793772, 0.793993, 0.794214}
```

In[658]:=
```
er = u − x;
er.er
```

Out[659]=
```
0.00215105
```

A Convergence result

For solving a linear system

$$A\,x = b$$

We use a stationary iterative method with splitting matrix F so that in the fixed-point fashion the iteration looks like:

$$F\,x^{(i+1)} = b - (A - F)\,x^{(i)}\,. \quad (1)$$

We know that convergence is granted whenever either one of the spectralradii

$$\rho\left(I - F^{-1} A\right) \quad \text{or} \quad \rho\left(I - A F^{-1}\right)$$

is less than one.

Without proof we state the following Theorem

Theorem

If stationary fixed-point iteration (1) is performed and both matrices A and $F + F^T - A$ are symmetric positive definite then $\rho(I - F^{-1} A) < 1$.

For a proof see for instance: A. Iserles; A first course in the Numerical Analysis of Differential Equations.

As a consequence we conclude that for any positive definite matrix, Gauss Seidel is convergent.

This can be seen as follows:

Assume that $A = L + D + L^T$ is *a* positive definite matrix with strictly lower triangular part L and diagonal D. The splitting matrix for Gauss-Seidel is either $(L + D)$ or $(D + L^T)$. In that case we have $F + F^T - A = D$ which is indeed symmetric positive definite.

Boosting the performance

We have seen Jacobi's and Gauss Seidel's method as examples of stationary iterative methods.

Recall the description of a general defect correction method.

Take some initial guess $x^{(0)}$ (the zero-vector is O.K.)

For $i = 0, 1, 2,$ until convergence DO :

Calculate: $r^{(i)} = b - A x^{(i)}$ ("defect")

Solve for $d^{(i)}$: $F d^{(i)} = r^{(i)}$ ("correction")

Add: $x^{(i+1)} = x^{(i)} + d^{(i)}$ ("adjustment")

Observation:

For the calculation of the next residual vector we must compute

$$r^{(i+1)} = b - A x^{(i+1)} = b - A (x^{(i)} + d^{(i)}) = r^{(i)} - A d^{(i)}$$

So we may construct the next residual vector from the current one:

$$r^{(i+1)} = r^{(i)} - A d^{(i)}$$

Now reformulate the defect correction method in a way to use

this recursive formulation.

Take some initial guess $x^{(0)}$ and calculate $r^{(0)} = b - A x^{(0)}$

For $i = 0, 1, 2,$ until convergence DO :

Solve: $F d^{(i)} = r^{(i)}$

Calculate: $c^{(i)} = A d^{(i)}$

Adjust: $r^{(i+1)} = r^{(i)} - c^{(i)}$

Add: $x^{(i+1)} = x^{(i)} + d^{(i)}$ ("adjustment")

(Convince your self that this is still a description of exactly the same defect correction method.)

Now comes the crucial part.

Like in SOR, it might be favourable to correct $x^{(i)}$ with a somewhat larger correction than $d^{(i)}$.

Keep in mind that $d^{(i)}$ controls the size and direction of $r^{(i+1)}$ via the quantity $c^{(i)}$ (= $A\,d^{(i)}$).

Because of the linearity of matrixmultiplication, we know that for any scalar ρ we have $\rho c^{(i)} = A\,\rho d^{(i)}$.

What about a reasonable choice for ρ in $x^{(i+1)} = x^{(i)} + \rho\, d^{(i)}$?

We can't know! But for any ρ being the coefficient of $d^{(i)}$, we should use the same ρ as coefficient for $c^{(i)}$ and therefore should construct the next residual from $r^{(i+1)} = r^{(i)} - \rho\, c^{(i)}$.

And now we can suggest a good value for ρ !

We know that convergence of $x^{(i)}$ to its limiting value, should go hand in hand with $||r^{(i)}||$ going to zero. So it makes sense to choose ρ such that $|| r^{(i+1)} ||$ is minimized and therefore we propose to calculate ρ such that $(|| r^{(i)} - \rho\, c^{(i)} ||)_2$ is minimal.

Omitting the index i, we find for the square of this 2-norm:

$$\rho^2\, c^T\, c - 2\,\rho\, c^T\, r + r^T\, r$$

which is quadratic in the variable ρ ; its minimal value is attained for

$$\rho = c^T\, r \;/\; c^T\, c$$

This choice is used as a strategy and implemented in the next examples of modified Jacobi and modified Gauss-Seidel. Use of this strategy is compared with the standard implementations.

▽ What about $c^T r = 0$?

If the quantity $c^T r$ should become zero, without any of the vectors c or r being zero (so c and r orthogonal), then we would have 'stagnation' .

In that unfortunate case we would have $x^{(i+1)} = x^{(i)}$ and also $r^{(i+1)} = r^{(i)}$ which tells us that from that moment on, none of the vectors $x^{(i)}$ and $r^{(i)}$ would change anymore, which may certainly be called *stagnation*.

Can it happen?

To analyse this situation, express c_i in terms of r_i.

From its definition we find

$$c_i = A\,d_i = A\,F^{-1}\,r_i$$

from which we find

$$c_i^T\, r_i = r_i^T\, c_i = r_i^T\, A\,F^{-1}\, r_i$$

Theorem:

If B is a positive definite matrix, then $x^T B\,x > 0$ for any vector $x \neq 0$.

So for an important class of matrices we can formulate the following statement:

For matrices A and splitting matrix F such that $A F^{-1}$ is positive definite, the variant of Jacobi which here is called Turbo Jacobi, shows no stagnation.

Theorem:

If A and F are symmetric positive definite then $A F^{-1}$ is positive definite.

Proof:

$x^T A F^{-1} x = x^T F^{-1/2} F^{+1/2} A F^{-1/2} F^{-1/2} x = y^T F^{+1/2} A F^{-1/2} y$ with $y = F^{-1/2} x$.

A and $F^{+1/2} A F^{-1/2}$ are similar matrices which have the same eigenvaluess.

This implies that $F^{+1/2} A F^{-1/2}$ is also positive definite from which the proof is clear.

Remark:

It is certainly not true that for any matrix B we have $x^T B x > 0$; a counterexample being:

$$B = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \text{ and } x = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

▽ Modified implementations:

Consider the same problem as before and use the same matrix A and right-hand side vector b.

▽ Jacobi's iterative algorithm in different formulation:

In[660]:=

```
x = {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.};
r = b;
i = 0;
res2 = 1;
```

In[664]:=

```
While[res2 > 0.01 && i < 50,
      res2 = r.r;
      d = LinearSolve[DD, r];
      c = A.d;
      r = r – c;
      x = x + d;
      i = i + 1;
  ];
res2
i
x
```

Out[665]=

```
0.00834881
```

Out[666]=

```
23
```

Out[667]=

```
{0.164043, 0.164043, 0.164043, 0.164043, 0.164043,
 0.341806, 0.341806, 0.341806, 0.341806, 0.341806,
 0.541787, 0.541787, 0.541787, 0.541787, 0.541787,
 0.764013, 0.764013, 0.764013, 0.764013, 0.764013}
```

In[668]:=

```
er = u – x;
er.er
```

Out[669]=

```
0.0468166
```

▽ **Modified Jacobi iterative algorithm:**

TURBO JACOBI

In[670]:=

```
x = {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.};
r = b;
i = 0;
res2 = 1;
```

In[674]:=

```
While[res2 > 0.01 && i < 50,
      res2 = r.r;
      d = LinearSolve[DD, r];
      c = A.d;
      rho = c.r / c.c;
      r = r – rho c;
      x = x + rho d;
      i = i + 1;
   ];
res2
i
x
```

Out[675]=

```
0.00837551
```

Out[676]=

```
12
```

Out[677]=

```
{0.170845, 0.170845, 0.170845, 0.170845, 0.170845,
 0.349127, 0.349127, 0.349127, 0.349127, 0.349127,
 0.552826, 0.552826, 0.552826, 0.552826, 0.552826,
 0.768558, 0.768558, 0.768558, 0.768558, 0.768558}
```

```
er = u – x;
er.er
```

▽ Implementation of Gauss-Seidel's iterative algorithm:

Here we only use the $U + D$ variant (= backward relaxation):

In[678]:=
```
x = {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.};
i = 0;
res2 = 1;
```

In[681]:=
```
While[res2 > 0.01 && i < 50,
    r = b - A.x;
    res2 = r.r;
    i = i + 1;
    d = LinearSolve[U + DD, r];
    x = x + d;
  ];
res2
i
x
```

Out[682]=
```
0.00889246
```

Out[683]=
```
12
```

Out[684]=
```
{0.176241, 0.175279, 0.174277, 0.173234, 0.17215,
 0.357537, 0.355816, 0.354025, 0.352161, 0.350221,
 0.55309, 0.551188, 0.549207, 0.547146, 0.545,
 0.767969, 0.766668, 0.765314, 0.763904, 0.762437}
```

In[685]:=
```
er = u - x;
er.er
```

Out[686]=
```
0.0330004
```

▽ **Modified Gauss Seidel iterative algorithm:**

TURBO GAUSS SEIDEL

In[687]:=

```
x = {0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0., 0.};
r = b;
i = 0;
res2 = 1;
```

In[691]:=

```
While[res2 > 0.01 && i < 50,
      res2 = r.r;
      d = LinearSolve[DD + U, r];
      c = A.d;
      rho = c.r / c.c;
      r = r – rho c;
      x = x + rho d;
      i = i + 1;
   ];
res2
i
x
```

Out[692]=

```
0.00734366
```

Out[693]=

```
7
```

Out[694]=

```
{0.186221, 0.185569, 0.185153, 0.184372, 0.184011,
 0.375305, 0.374319, 0.373295, 0.371926, 0.371345,
 0.572753, 0.571394, 0.57052, 0.568435, 0.568756,
 0.781225, 0.780118, 0.780339, 0.778029, 0.777982}
```

In[695]:=

```
er = u – x;
er.er
```

Out[696]=

```
0.0112144
```

Close all sections