MTH 9821 Homework Two *

Chu, Hongshan
Wang, Jiaxi
Wei, Zhaoyue
Yin, Gongshun
Zhou, ShengQuan

September 19, 2016

*Team work:
Chu, Hongshan #10, #14
Wang, Jiaxi #12
Wei, Zhaoyue #13
Yin, Gongshun #2,#3,#4,#5,#6,#7
Zhou, ShengQuan #1,#2,#3,#4,#5,#6,#7,#8,#9,#11,#13

1 C++ Codes

1.1 Cholesky Decomposition of an S.P.D. Matrix, Banded, Tridiagonal

```
Eigen::MatrixXd cholesky(const Eigen::MatrixXd & A)
{
    int n = A.rows();
    return cholesky_banded(A, n-1);
}

Eigen::MatrixXd cholesky_banded(const Eigen::MatrixXd & A, int m)
{
    Eigen::MatrixXd Acopy = A;
    int n = Acopy.rows();
    assert(n == Acopy.cols());
    assert(Acopy == Acopy.transpose()); // enfoce symmetry

    Eigen::MatrixXd U = Eigen::MatrixXd::Zero(n,n);

    for (int k=0; k<n-1; k++) {
        assert(Acopy(k,k)>0);

        U(k,k) = std::sqrt(Acopy(k,k));

        int boundary = std::min(k+m+1,n);
        for (int i=k+1; i<boundary; i++) {
            U(k,i) = Acopy(k,i)/U(k,k);
        }

        for (int i=k+1; i<boundary; i++) {
            for (int j=i; j<boundary; j++) {
                Acopy(i,j) -= U(k,i)*U(k,j);
            }
        }
    }

    U(n-1,n-1) = std::sqrt(Acopy(n-1,n-1));

    return U;
}

Eigen::MatrixXd cholesky_tridiagonal(const Eigen::MatrixXd & A)
{
    return cholesky_banded(A, 1);
}
```

```
Eigen::VectorXd spd_solve(const Eigen::MatrixXd & A,
                          const Eigen::VectorXd & b)
{
    int n = b.size();
    assert(A.rows() == n);
    assert(A.cols() == n);

    Eigen::MatrixXd U = cholesky(A);
    Eigen::VectorXd y = forward_subst(U.transpose(), b);
    Eigen::VectorXd x = backward_subst(U, y);

    return x;
}

Eigen::VectorXd banded_spd_solve(const Eigen::MatrixXd & A, int m,
                                 const Eigen::VectorXd & b)
{
    int n = b.size();
    assert(A.rows() == n);
    assert(A.cols() == n);

    Eigen::MatrixXd U = cholesky_banded(A, m);
    Eigen::VectorXd y = forward_subst_banded(U.transpose(), m, b);
    Eigen::VectorXd x = backward_subst_banded(U, m, y);

    return x;
}

Eigen::VectorXd tridiagonal_spd_solve(const Eigen::MatrixXd & A,
                                      const Eigen::VectorXd & b)
{
    int n = b.size();
    assert(A.rows() == n);
    assert(A.cols() == n);

    Eigen::MatrixXd U = cholesky_tridiagonal(A);
    Eigen::VectorXd y = forward_subst_tridiagonal(U.transpose(), b);
    Eigen::VectorXd x = backward_subst_tridiagonal(U, y);

    return x;
}
```

where the methods for triangular solvers are defined in the next sub-section.

```
Eigen::VectorXd forward_subst_banded(const Eigen::MatrixXd & L, int m,
                                     const Eigen::VectorXd & b)
{
    int n = b.size();
    assert(L.rows() == n);
    assert(L.cols() == n);

    Eigen::VectorXd x = Eigen::VectorXd::Zero(n);
    x(0) = b(0)/L(0,0);
    for (int i=1; i<n; i++) {
        double sum = 0;
        int boundary = std::max(0,i-m);
        for (int j=boundary; j<i; j++) {
            sum += L(i,j)*x(j);
        }
        x(i) = (b(i)-sum)/L(i,i);
    }

    return x;
}

Eigen::VectorXd forward_subst(const Eigen::MatrixXd & L,
                              const Eigen::VectorXd & b)
{
    int n = b.size();
    return forward_subst_banded(L, n-1, b); //band width = n-1
}

Eigen::VectorXd forward_subst_tridiagonal(const Eigen::MatrixXd & L,
                                          const Eigen::VectorXd & b)
{
    return forward_subst_banded(L, 1, b); //band width = 1
}
```

```
Eigen::VectorXd backward_subst_banded(const Eigen::MatrixXd & U, int m,
                                      const Eigen::VectorXd & b)
{
    int n = b.size();
    assert(U.rows() == n);
    assert(U.cols() == n);

    Eigen::VectorXd x = Eigen::VectorXd::Zero(n);
    x(n-1) = b(n-1)/U(n-1,n-1);
    for (int i=n-2; i>=0; i--) {
        double sum = 0;
        int boundary = std::min(n,i+m+1);
        for (int j=i+1; j<boundary; j++) {
            sum += U(i,j)*x(j);
        }
        x(i) = (b(i)-sum)/U(i,i);
    }

    return x;
}

Eigen::VectorXd backward_subst(const Eigen::MatrixXd & U,
                               const Eigen::VectorXd & b)
{
    int n = b.size();
    return backward_subst_banded(U, n-1, b); //band width = n-1
}

Eigen::VectorXd backward_subst_tridiagonal(const Eigen::MatrixXd & U,
                                           const Eigen::VectorXd & b)
{
    return backward_subst_banded(U, 1, b); //band width = 1
}
```

2 Verification of Symmetric Positive Definiteness

Let

$$A = \begin{pmatrix} 1 & -0.2 & 0.8 \\ -0.2 & 2 & 0.3 \\ 0.8 & 0.3 & 1.1 \end{pmatrix}$$

(i) Show that the matrix A is weakly diagonally dominated, and therefore symmetric positive semi-definite.
Proof: Firstly, A is symmetric by observation, thus all eigenvalues are real. Check each row:

$$\begin{aligned} |A_{11}| &= 1 = 0.2 + 0.8 = |A_{12}| + |A_{13}|, \\ |A_{22}| &= 2 > 0.2 + 0.3 = |A_{21}| + |A_{23}|, \\ |A_{33}| &= 1.1 = 0.8 + 0.3 = |A_{31}| + |A_{32}|. \end{aligned}$$

Therefore, the matrix A is weakly diagonally dominated and symmetric positive semi-definite.

(ii) Use Sylvester's criterion to show that the matrix A is symmetric positive definite.
Proof: Check each leading principal minor:

$$\begin{aligned} &A_{11} = 1 > 0, \\ &\det \begin{vmatrix} 1 & -0.2 \\ -0.2 & 2 \end{vmatrix} = 2 - 0.04 > 0, \\ &\det \begin{vmatrix} 1 & -0.2 & 0.8 \\ -0.2 & 2 & 0.3 \\ 0.8 & 0.3 & 1.1 \end{vmatrix} = 2.2 - 0.048 - 0.048 - 1.28 - 0.044 - 0.09 = 0.69 > 0. \end{aligned}$$

According to Sylvester's criterion, the matrix A is symmetric positive definite.

3 Cholesky Decomposition of a Special Matrix

Let B_N be the $N \times N$ tridiagonal symmetric positive definite matrix given by

$$B_N = \begin{pmatrix} 2 & -1 & \cdots & 0 \\ -1 & 2 & \cdots & 0 \\ \vdots & \vdots & \ddots & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix}.$$

Show that the Cholesky factor U_N of the matrix B_N is the upper triangular bidiagonal matrix given by

$$\begin{aligned} U_N(i,i) &= \sqrt{\frac{i+1}{i}}, \quad \forall i = 1, \cdots, N, \\ U_N(i,i+1) &= -\sqrt{\frac{i}{i+1}}, \quad \forall i = 1, \cdots, N-1. \end{aligned} \tag{3.1}$$

Proof: Write the elements of matrix B_N and U_N as

$$\begin{aligned} B_N(i,j) &= 2\delta_{ij} - \delta_{i,j+1} - \delta_{i,j-1}, \\ U_N(i,j) &= \sqrt{\frac{i+1}{i}}\delta_{ij} - \sqrt{\frac{i}{i+1}}\delta_{i+1,j}. \end{aligned}$$

Compute the matrix mulplication, $\forall i, j = 1, \cdots, N$

$$\begin{aligned} \left(U_N^T U_N\right)(i,j) &= \sum_{k=1}^{N} U_N^T(i,k) U_N(k,j) \\ &= \sum_{k=1}^{N} \left(\sqrt{\frac{k+1}{k}}\delta_{ki} - \sqrt{\frac{k}{k+1}}\delta_{k+1,i} \right) \left(\sqrt{\frac{k+1}{k}}\delta_{kj} - \sqrt{\frac{k}{k+1}}\delta_{k+1,j} \right) \\ &= \sum_{k=1}^{N} \left(\frac{k+1}{k}\delta_{ki}\delta_{kj} - \delta_{k+1,i}\delta_{kj} - \delta_{ki}\delta_{k+1,j} + \frac{k}{k+1}\delta_{k+1,i}\delta_{k+1,j} \right) \\ &= \frac{i+1}{i}\delta_{ij} - \delta_{j+1,i} - \delta_{i+1,j} + \frac{i-1}{i}\delta_{ij} \\ &= 2\delta_{ij} - \delta_{i,j+1} - \delta_{i,j-1} \\ &= B_N(i,j). \end{aligned}$$

Since we know that B_N is spd matrix so it has Cholesky decomposition and it is unique. Hence the U_N described here must be the Cholesky factor of matrix B_N.

4 Pseudocode of Cholesky Linear Solver for a Special Matrix

Let B_N and U_N be the matrices given in the previous problem.
(i) Give the pseudocode for computing the solution to a linear system $B_N x = b$, where b and x are $N \times 1$ column vectors.
Solution: Solving the linear system $B_N x = b$ is equivalent to solving the following two linear systems successively

$$U_N^T y = b,$$
$$U_N x = y.$$

Using the explicit matrix elements of U_N given in the previous problem, we can write down the pseudocode

Algorithm 1 Cholesky Linear Solve of B_N

Input: $b : N \times 1$ column vector
Output: x : solution to $B_N x = b$
$y_1 = \frac{b_1}{U_{11}} = \frac{b_1}{\sqrt{2}}$
For $i = 2 : N$
$$y_i = \frac{b_i - U_{i-1,i} y_{i-1}}{U_{ii}} = \frac{b_i + y_{i-1}\sqrt{\frac{i-1}{i}}}{\sqrt{\frac{i+1}{i}}}$$
End
$x_N = \frac{y_N}{U_{NN}} = y_N \sqrt{\frac{N}{N+1}}$
For $i = N-1 : 1$
$$x_i = \frac{y_i - U_{i,i+1} x_{i+1}}{U_{ii}} = \frac{y_i + x_{i+1}\sqrt{\frac{i}{i+1}}}{\sqrt{\frac{i+1}{i}}}$$
End

(ii) Give the operation count of the pseudocode above. How does it compare to $8n + O(1)$.
Solution: The operation count is

$$2 + (N-1) \times 9 + 5 + (N-1) \times 9 = 18N + O(1),$$

where the 9 operations per interation include addition/subtraction, multiplication/division, and square roots. Compared with $8n + O(1)$, the op count of this algorithm is larger.

5 *LU*-DECOMPOSITION OF A SPECIAL MATRIX

Let B_N be the tridiagonal matrix given in the previous two problems. Show that the *LU*-factors L and U of the matrix B_N are the lower triangular bidiagonal matrix and the upper triangular bidiagonal matrix given by

$$\begin{aligned}
&L(i,i) = 1, \quad \forall i = 1, \cdots, N \\
&L(i+1,i) = -\frac{i}{i+1}, \quad \forall i = 1, \cdots, N-1 \\
&U(i,i) = \frac{i+1}{i}, \quad \forall i = 1, \cdots, N \\
&U(i,i+1) = -1, \quad \forall i = 1, \cdots, N-1
\end{aligned}$$

Proof: Similarly, write the matrix elements explicitly

$$\begin{aligned}
L_{ij} &= \delta_{ij} - \frac{i-1}{i}\delta_{i,j+1}, \\
U_{ij} &= \frac{i+1}{i}\delta_{ij} - \delta_{i,j-1}.
\end{aligned}$$

Compute the matrix mulplication, $\forall i, j = 1, \cdots, N$

$$\begin{aligned}
(LU)_{ij} &= \sum_{k=1}^{N} L_{ik}U_{kj} \\
&= \sum_{k=1}^{N} \left(\delta_{ik} - \frac{i-1}{i}\delta_{i,k+1}\right)\left(\frac{k+1}{k}\delta_{kj} - \delta_{k,j-1}\right) \\
&= \sum_{k=1}^{N} \left(\frac{k+1}{k}\delta_{ik}\delta_{kj} - \delta_{ik}\delta_{k,j-1} - \frac{(i-1)(k+1)}{ik}\delta_{i,k+1}\delta_{kj} + \frac{i-1}{i}\delta_{i,k+1}\delta_{k,j-1}\right) \\
&= \frac{i+1}{i}\delta_{ij} - \delta_{i,j-1} - \delta_{i,j+1} + \frac{i-1}{i}\delta_{ij} \\
&= 2\delta_{ij} - \delta_{i,j+1} - \delta_{i,j-1} \\
&= B_N(i,j).
\end{aligned}$$

Since we know that B_N is spd matrix so it has *LU* decomposition and it is unique. Hence the L, U described here must be the L, U factors of matrix B_N.

6 PSEUDOCODE OF *LU* LINEAR SOLVER FOR A SPECIAL MATRIX

Let B_N, L, and U be the matrices given in the previous problem.
(i) Give the pseudocode for computing the solution to a linear system $B_N x = b$, where b and x are $N \times 1$ column vectors.
Solution: Solving the linear system $B_N x = b$ is equivalent to solving the following two linear systems successively

$$Ly = b,$$
$$Ux = y.$$

Using the explicit matrix elements of L and U given in the previous problem, we can write down the pseudocode

Algorithm 2 *LU* Linear Solve of B_N

Input: $b : N \times 1$ column vector
Output: x : solution to $B_N x = b$
$y_1 = \frac{b_1}{L_{11}} = b_1$
For $i = 2 : N$
    $y_i = \frac{b_i - L_{i,i-1} y_{i-1}}{L_{ii}} = b_i + y_{i-1} \frac{i-1}{i}$
End
$x_N = \frac{y_N}{U_{NN}} = y_N \frac{N}{N+1}$
For $i = N-1 : 1$
    $x_i = \frac{y_i - U_{i,i+1} x_{i+1}}{U_{ii}} = \frac{i}{i+1} \left(y_i + x_{i+1} \right).$
End

(ii) Give the operation count of the pseudocode above. How does it compare to $8n + O(1)$.
Solution: The operation count is

$$(N-1) \times 4 + 3 + (N-1) \times 4 = 8N + O(1),$$

where the 4 operations per interation include addition/subtraction and multiplication/division. Hence the algorithm has the same op count as $8n + O(1)$.

7 Optimal Pseudocode for Tridiagonal S.P.D. Solver

Write an explicit optimal pseudocode for solving linear systems corresponding to the same tridiagonal symmetric positive definite matrix. In other words, write a psedudocode for solving p linear systems $Ax_i = b_i$, $\forall i = 1:p$, where A is an $n \times n$ tridiagonal symmetric positive definite matrix.

Algorithm 3 *LU* Linear Solve of Tridiagonal Symmetric Positive Definite Matrix

Input:
$A : N \times N$ tridiagonal symmetric positive definite matrix
$b^i : N \times 1$ column vectors, $i = 1, \cdots, p$
Output: $x^i : N \times 1$ column vectors, solution to $Ax^i = b^i$, $i = 1, \cdots, p$

$L = 0, U = 0$
For $i = 1 : N-1$
  $L_{ii} = 1$
  $U_{ii} = A_{ii}$
  $U_{i,i+1} = A_{i,i+1}$
  $L_{i+1,i} = \frac{A_{i+1,i}}{A_{ii}}$
  $A_{i+1,i+1} = A_{i+1,i+1} - L_{i+1,i}U_{i,i+1}$
End
$L_{NN} = 1, U_{NN} = A_{NN}$

For $i = 1 : p$
  $y_1 = b_1^i$
  For $j = 2 : N$
    $y_j = \frac{b_j^i - L_{j,j-1}y_{j-1}}{L_{jj}}$
  End

  $x_N^i = \frac{y_N}{U_{NN}}$
  For $j = N-1 : 1$
    $x_j^i = \frac{y_j - U_{j,j+1}x_{j+1}^i}{U_{jj}}$
  End
End

$$\begin{aligned}\text{Operation Count} &= (n-1)\cdot 3 + p\cdot((n-1)\cdot 2 + 1 + (n-1)\cdot 3)\\ &= 3n - 3 + p\cdot(5n-4)\\ &= 5np - 4p + 3n - 3.\end{aligned}$$

8 Pseudocode for Cholesky Decomposition of Banded S.P.D.

Write the pseudocode for the Cholesky decomposition of symmetric positive definite banded matrices of band m. What is the corresponding operation count?

Algorithm 4 Cholesky Decomposition of Banded S.P.D.

Input: $A : N \times N$ banded symmetric positive definite matrix with band width m
Output: $U : N \times N$ upper triangular matrix $U^T U = A$

$U = 0$
For $k = 1 : N-1$
  $U_{kk} = \sqrt{A_{kk}}$
  For $i = k+1 : \min(k+m, n)$
    $U_{ki} = \frac{A_{ki}}{U_{kk}}$
  End

  For $i = k+1 : \min(k+m, n)$
    For $j = i : \min(k+m, n)$
      $A_{i,j} = A_{i,j} - U_{k,i} U_{k,j}$
      $A_{j,i} = A_{i,j}$
    End
  End
End
$U_{NN} = \sqrt{A_{NN}}$

The operation count

$$
\begin{aligned}
&(n-1) + m \times (n-m) + \frac{m(m-1)}{2} + \frac{m(m+1)}{2} \times 2(n-m) + \frac{m(m-1)(m+1)}{3} + 1 \\
&= \left(m^2 + 2m + 1\right) n - \frac{2}{3} m^3 + O\left(m^2\right).
\end{aligned}
$$

9 Cholesky Linear Solve of Banded S.P.D.

What is the operation count for solving a linear system corresponding to a symmetric positive definite banded matrix of band m using a Cholesky linear solver?
Solution: The Cholesky linear solve for $U^T Ux = b$ is carried out in the following two steps in succession:

$$U^T y = b,$$
$$Ux = y.$$

We write down the pseudocode

Algorithm 5 Cholesky Linear Solve of Banded Symmetric Positive Definite Matrix

Input:
Cholesky decomposition of an S.P.D. matrix with band width m: $A = U^T U$
$b : N \times 1$ column vector
Output: x : solution to $U^T Ux = b$

$y_1 = \frac{b_1}{U_{11}}$
For $i = 2 : n$
sum=0
For $j = \max(0, i-m) : (i-1)$
sum = sum + $U_{ji} y_j$
End
$y_i = \frac{b_i - \text{sum}}{U_{ii}}$
End

$x_n = \frac{y_n}{U_{nn}}$
For $i = n-1 : 1$
sum=0
For $j = i+1 : \min(n, i+m+1)$
sum = sum + $U_{ij} x_j$
End
$x_i = \frac{y_i - \text{sum}}{U_{ii}}$
End

The operation count corresponding to the above pseudocode for the Cholesky linear solve of banded symmetric positive definite matrix is given in Problem 4 and Problem 5 in Homework One:

$$2 \times (2mn - m^2) + O(m) = 4mn - 2m^2 + O(m).$$

Including the operation count corresponding to the Cholesky decomposition from the previous problem, we obtain the total operation count for solving one linear system

$$\left(m^2 + 2m + 1\right) n - \frac{2}{3} m^3 + O\left(m^2\right) + \left(4mn - 2m^2\right) + O(m) = \left(m^2 + 6m + 1\right) n - \frac{2}{3} m^3 + O\left(m^2\right).$$

10 CUBIC SPLINE INTERPOLATION OF ZERO RATES

The following discount factors were obtained from market data:

Date	Discount Factor
2 months	0.9980
5 months	0.9935
11 months	0.9820
15 months	0.9775

The overnight rate is 0.75%.

(i) What are the corresponding 2 months, 5 months, 11 months, and 15 months zero rates?
Solution: Since

$$\text{disc}(t) = e^{-r(0,t)t}$$

we have

$$r(0,t) = \frac{\ln(\text{disc}(t))}{-t}$$

thus we have

$$\begin{aligned} r(0,0) &= 0.0075, \\ r\left(0,\frac{2}{12}\right) &= 0.01201202, \\ r\left(0,\frac{5}{12}\right) &= 0.01565092, \\ r\left(0,\frac{11}{12}\right) &= 0.01981524, \\ r\left(0,\frac{15}{12}\right) &= 0.01820559. \end{aligned}$$

In tabular form,

Date	Discount Factor	Zero Rate
2 months	0.9980	0.01201202
5 months	0.9935	0.01565092
11 months	0.9820	0.01981524
15 months	0.9775	0.01820559

(ii) What is the tridiagonal system that must be solved in the efficient implementation of the natural cubic spline interpolation for finding the zero rate curve for all times less than 15 months?
Solution: Following the pseudocode in Table 6.8 in the textbook, we need solve the tridiagonal system:

$$W = \begin{pmatrix} 5/6 & 1/4 & 0 \\ 1/4 & 3/2 & 1/2 \\ 0 & 1/2 & 5/3 \end{pmatrix}, \quad z = \begin{pmatrix} 0.0750988 \\ -0.037361875 \\ -0.07894557 \end{pmatrix}$$

and we have to solve the linear system

$$Mw = z.$$

(iii) Use the efficient implementation of the natural cubic spline interpolation to find a zero rate curve for all times less than 15 months matching the discount factor above.
Solution: Use the C++ implementation of linear solver:

$$r(0,t) \approx \begin{cases} 0.0075 + 0.02963324\,t - 0.09220133\,t^2, & 0 \le t \le \frac{2}{12} \\ 0.006767143 + 0.04282467\,t - 0.079148546\,t^2 + 0.066095763\,t^2, & \frac{2}{12} \le t \le \frac{5}{12} \\ 0.012908145 - 0.001390545\,t + 0.02096797\,t^2 - 0.01879448\,t^3, & \frac{5}{12} \le t \le \frac{11}{12} \\ -0.020615233 + 0.1083223\,t - 0.0927188\,t^2 + 0.024725014\,t^3, & \frac{11}{12} \le t \le \frac{15}{12} \end{cases}$$

(iv) Find the value of a 14-months quarterly coupon bond with 2.5% coupon rate.
Solution:

$$\begin{aligned} PV &= 0.625 \cdot \mathrm{disc}\left(\frac{2}{12}\right) + 0.625 \cdot \mathrm{disc}\left(\frac{5}{12}\right) + 0.625 \cdot \mathrm{disc}\left(\frac{8}{12}\right) + 0.625 \cdot \mathrm{disc}\left(\frac{11}{12}\right) + 100.625 \cdot \mathrm{disc}\left(\frac{14}{12}\right) \\ &\approx 100.915 \end{aligned}$$

11 Find Normal Variables with Given Covariance Matrix

Given Z_1, Z_2, Z_3 independent standard normal variables, find three three normal random variables X_1, X_2, X_3 with covariance matrix

$$\begin{pmatrix} 1 & 1 & 0.5 \\ 1 & 4 & -2 \\ 0.5 & -2 & 9 \end{pmatrix}.$$

Solution: Perform the Cholesky decomposition of the given covariance matrix

$$\begin{pmatrix} 1 & 1 & 0.5 \\ 1 & 4 & -2 \\ 0.5 & -2 & 9 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0.5 \\ 0 & 1.732051 & -1.443376 \\ 0 & 0 & 2.581989 \end{pmatrix}^T \begin{pmatrix} 1 & 1 & 0.5 \\ 0 & 1.732051 & -1.443376 \\ 0 & 0 & 2.581989 \end{pmatrix}$$

Thus,

$$\begin{aligned} X_1 &= Z_1, \\ X_2 &= Z_1 + 1.732051 Z_2, \\ X_3 &= 0.5 Z_1 - 1.443376 Z_2 + 2.581989 Z_3. \end{aligned}$$

12 Estimation of Implied Volatility

On May 22, 2014, the bid and ask prices of the S&P 500 options maturing on January 17, 2015 were given. The spot prices of the index corresponding to these option prices was 1,894.

(i) Compute the mid prices of the options, i.e., the average of the bid price and ask price of the options. Use ordinary least squares to compute the present value of the forward price PVF and the discount factor disc corresponding to the mid prices of the options.
Solution: The mid prices are listed in the following table

Strike	Call Bid Price	Call Ask Price	Call Mid Price	Put Bid Price	Put Ask Price	Put Mid Price	Call-Put
1450	431.2	434.4	432.8	8.9	10.2	9.55	423.25
1500	384.1	387.4	385.75	11.6	13.1	12.35	373.4
1550	337.9	341.2	339.55	15.2	16.9	16.05	323.5
1600	292.8	296.2	294.5	19.9	21.8	20.85	273.65
1675	228.3	231.1	229.7	29.7	31.4	30.55	199.15
1700	207.5	210.1	208.8	33.6	35.5	34.55	174.25
1750	167.6	170	168.8	43.5	45.4	44.45	124.35
1775	148.9	151.3	150.1	49.4	51.5	50.45	99.65
1800	130.6	132.8	131.7	56.5	58.1	57.3	74.4
1825	113.2	115.3	114.25	63.6	65.7	64.65	49.6
1850	96.8	98.6	97.7	71.9	74.2	73.05	24.65
1875	81.5	83.2	82.35	81.4	83.5	82.45	-0.1
1900	67.1	69	68.05	92.1	94.5	93.3	-25.25
1925	54.2	55.9	55.05	103.8	106.3	105.05	-50
1975	32.8	34.5	33.65	132.4	134.7	133.55	-99.9
2000	24.5	25.9	25.2	149.2	151.9	150.55	-125.35
2050	12.4	13.6	13	186.9	189.7	188.3	-175.3
2100	5.6	6.5	6.05	230	232.4	231.2	-225.15

The problem is equivalent to finding the least square solution to $\bar{C} - \bar{P} = \text{PVF} - K \times \text{disc}$:

$$\begin{pmatrix} 1 & -1450 \\ 1 & -1500 \\ 1 & -1550 \\ 1 & -1600 \\ 1 & -1675 \\ 1 & -1700 \\ 1 & -1750 \\ 1 & -1775 \\ 1 & -1800 \\ 1 & -1825 \\ 1 & -1850 \\ 1 & -1875 \\ 1 & -1900 \\ 1 & -1925 \\ 1 & -1975 \\ 1 & -2000 \\ 1 & -2050 \\ 1 & -2100 \end{pmatrix} \times \begin{pmatrix} \text{PVF} \\ \text{disc} \end{pmatrix} \approx \begin{pmatrix} 423.25 \\ 373.4 \\ 323.5 \\ 273.65 \\ 199.15 \\ 174.25 \\ 124.35 \\ 99.65 \\ 74.4 \\ 49.6 \\ 24.65 \\ -0.1 \\ -25.25 \\ -50 \\ -99.9 \\ -125.35 \\ -175.3 \\ -225.15 \end{pmatrix}$$

The least square solution gives

$$\text{PVF} \approx 1869.403121, \quad \text{disc} \approx 0.997227746.$$

(ii) Compute the implied volatilities of these options. How do the implied volatilities of calls and puts with the same strike compare to each other?
Solution: Implied volatility σ is defined by Black-Scholes formula:

$$C = +\text{PVF} \cdot N(+d_+) - K \cdot \text{disc} \cdot N(+d_-),$$
$$P = -\text{PVF} \cdot N(-d_+) + K \cdot \text{disc} \cdot N(-d_-),$$

where $N(x)$ is the cumulative standard normal function,

$$d_\pm = \frac{\ln\left(\frac{\text{PVF}}{K \cdot \text{disc}}\right)}{\sigma\sqrt{T}} \pm \frac{\sigma\sqrt{T}}{2},$$

and $T = \frac{166}{252}$. Solve the above equation numerically using market data, we get

Strike	Call Implied Vol (10^{-2})	Put Implied Vol (10^{-2})	Percentage Difference
1450	21.87	21.97	-0.46%
1500	20.89	20.96	-0.34%
1550	19.93	20.00	-0.35%
1600	19.00	19.06	-0.32%
1675	17.64	17.61	+0.17%
1700	17.14	17.11	+0.18%
1750	16.16	16.14	+0.12%
1775	15.73	15.67	+0.38%
1800	15.21	15.21	+0.00%
1825	14.71	14.69	+0.14%
1850	14.21	14.19	+0.14%
1875	13.73	13.68	+0.36%
1900	13.23	13.21	+0.15%
1925	12.73	12.69	+0.31%
1975	11.83	11.79	+0.34%
2000	11.40	11.47	-0.61%
2050	10.67	10.78	-1.03%
2100	10.12	10.28	-1.58%

The percentage difference is defined as the different between the call implied volatility and the put implied volatility relative to the call implied volatility. From the above result, we observe that a good agreement between the call implied volatility and the put implied volatility is obtained.

13 Comparison between Least Square, Linear Interpolation, and Cubic Spline Interpolation

Recall the example from the book where for 15 consecutive trading days, the yields of the 2-year, 3-year, 5-year, and 10-year treasury bonds were, respectively:

2-year	3-year	5-year	10-year	3-year least square	3-year linear interpolated	3-year cubic spline
4.69	4.58	4.57	4.63	4.588379	4.650000	4.641333
4.81	4.71	4.69	4.73	4.696707	4.770000	4.762000
4.81	4.72	4.70	4.74	4.705345	4.773333	4.765889
4.79	4.78	4.77	4.81	4.763268	4.783333	4.780889
4.79	4.77	4.77	4.80	4.757971	4.783333	4.781222
4.83	4.75	4.73	4.79	4.744399	4.796667	4.789111
4.81	4.71	4.72	4.76	4.722621	4.780000	4.773667
4.81	4.72	4.74	4.77	4.734600	4.786667	4.781778
4.83	4.76	4.77	4.80	4.763059	4.810000	4.805667
4.81	4.73	4.75	4.77	4.737941	4.790000	4.786000
4.82	4.75	4.77	4.80	4.761787	4.803333	4.799556
4.82	4.75	4.76	4.80	4.758446	4.800000	4.795333
4.80	4.73	4.75	4.78	4.741966	4.783333	4.779556
4.78	4.71	4.72	4.73	4.702912	4.760000	4.756333
4.79	4.71	4.71	4.73	4.700843	4.763333	4.758222

Denote by T_2, T_3, T_5, and T_{10} the time series data vectors corresponding to the yield of the 2-year, 3-year, 5-year, and 10-year treaury bonds, respectively.
(i) Find the coefficients a, b_1, b_2, b_3 of the linear regression for the yield of the 3-year bond in terms of the yields of the 2-year, 5-year, and 10-year bonds, i.e., find a, b_1, b_2, b_3 corresponding to the solution to the ordinary least squares problem

$$T_3 \approx a\mathbf{1} + b_1 T_2 + b_2 T_5 + b_3 T_{10},$$

where $\mathbf{1}$ is the 15×1 column vector with all entries equal to 1. Let

$$T_{3,LR} = a\mathbf{1} + b_1 T_2 + b_2 T_5 + b_3 T_{10}.$$

Find the approxmation error

$$\text{error}_{LR} \triangleq \| T_3 - T_{3,LR} \|.$$

Solution: Let

$$A = (\mathbf{1}, T_1, T_5, T_{10}), \quad x = \begin{pmatrix} a \\ b_1 \\ b_2 \\ b_3 \end{pmatrix}, \quad y = T_3.$$

We are looking for the least square solution to

$$Ax \approx y.$$

In other words,

$$A^T Ax = A^T y.$$

Use the C++ code implementation of the linear solve for S.P.D. matrix, we get the following numerical result

$$x = \begin{pmatrix} a \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} 0.01232 \\ 0.127208 \\ 0.334045 \\ 0.529777 \end{pmatrix}.$$

The approximation error

$$\text{error}_{LR} \triangleq \| T_3 - T_{3,LR} \| \approx 0.043.$$

(ii) Compute the linear interpolation values of the 3-year yield by doing linear interpolation between the 2-year yield and the 5-year yield at each data point. Denote by

$$T_{3,\text{linear interp}} \triangleq \frac{2}{3} T_2 + \frac{1}{3} T_5.$$

Find the approximation error

$$\text{error}_{\text{linear interp}} = \| T_3 - T_{3,\text{linear interp}} \|$$

of the linear interpolation.
Solution: The three year interpolated values are listed in the table given in the problem statement. The approximation error

$$\text{error}_{\text{linear interp}} = \| T_3 - T_{3,\text{linear interp}} \| \approx 0.207.$$

(iii) Compute the cubic interpolation values of the 3-year yield by doing cubic spline interpolation between the 2-year, 5-year, and 10-year yield at each data point. Denote by $T_{3,\text{cubic interp}}$ the time series vector of these values. Find the approximation error

$$\text{error}_{\text{cubic interp}} = \| T_3 - T_{3,\text{cubic interp}} \|$$

of the cubic spline interpolation.
Solution: Basically, we are doing cubic spline interpolation over the domain

$$x = \big(2, 5, 10\big)$$

for 15 independent datasets

$$y^{(i)} = \big(T_2(i), T_5(i), T_{10}(i)\big), \quad \forall\, i = 1, \cdots, 15.$$

For example,

$$y^{(1)} = \big(4.69, 4.57, 4.63\big)$$
$$y^{(2)} = \big(4.81, 4.69, 4.73\big)$$
$$\cdots$$

The cubic spline interpolated values are listed in the table given in the problem statement. The approximation error

$$\text{error}_{\text{cubic interp}} = \| T_3 - T_{3,\text{cubic interp}} \| \approx 0.186.$$

(iv) The ordinary least square method gives the lowest error. Cubic spline interplation gives relatively lower error compared to linear interpolation method.

14 CHAPTER 8, EXERCISE 8

The file *financials2012.xlsx* from www.fepress.org/nla-primer contains the end of week adjusted closing prices for the stocks of the following financial companies: JPM; GS; MS; BAC; RBS; CS; UBS; RY; BCS between January 11, 2012 and October 15, 2012.

(i) Compute the weekly percentage returns of these stocks.
Solution: The percentage return table:

Date	JPM	GS	MS	BAC	RBS	CS	UBS	RY..RBC.	BCS..Barclays.
1/17/2012	0.04013664	0.09881342	0.10625380	0.06990882	0.16711957	0.15071003	0.14792899	0.04420114	0.12727273
1/23/2012	-0.00410509	0.02792776	0.01152580	0.03125000	0.01396973	0.02826433	0.02577320	-0.00485437	0.01026393
1/30/2012	0.02885408	0.05153052	0.09441129	0.07575758	0.04362801	0.04413473	0.04379038	0.02478049	0.07184325
2/6/2012	-0.01762821	-0.02902420	-0.03173029	0.02944942	-0.03520352	-0.07675195	-0.05226960	-0.00342727	-0.01895735
2/13/2012	0.02311039	0.01569984	-0.02560164	-0.00621891	0.00912201	0.04216868	0.03047896	-0.00133741	0.07039338
2/21/2012	-0.00504916	-0.00034931	-0.03468208	-0.01752190	0.02146893	0.03121387	-0.00211268	0.02353166	-0.00064475
2/27/2012	0.06143162	0.03843802	0.02014154	0.03312102	-0.02323009	-0.01382661	-0.03034580	0.04429907	0.03225807
3/5/2012	0.00981379	-0.02229326	-0.02614728	-0.00986437	-0.06681767	-0.03031451	-0.02838428	0.00501163	-0.06250000
3/12/2012	0.08621979	0.04809843	0.06301370	0.21668742	0.09101942	0.11645174	0.07191011	0.02137133	0.06666667
3/19/2012	0.01330580	0.02643461	0.04072165	0.00511771	-0.00222469	-0.01400070	-0.01397624	-0.00871840	-0.02125000
3/26/2012	0.01811184	-0.01431656	-0.03368004	-0.02749491	-0.01449275	-0.02378417	-0.01488306	-0.00035180	-0.04214559
4/2/2012	-0.02935290	-0.05128205	-0.06355715	-0.03560209	-0.09389140	-0.06545455	-0.06546763	-0.01372515	-0.08266667
4/9/2012	-0.02542955	-0.02463223	-0.06075534	-0.05971770	-0.01872659	-0.03424125	-0.04772902	-0.02123104	-0.02180233
4/16/2012	-0.01128350	-0.02297440	0.01165501	-0.03695150	-0.02671756	0.01329573	0.00161682	0.02825374	0.00445765
4/23/2012	0.01450309	0.01750135	-0.02764977	-0.01318945	0.03921569	-0.05487078	0.01614205	0.01861372	0.06360947
4/30/2012	-0.03679400	-0.04736703	-0.05568720	-0.06196841	-0.00880503	-0.08834666	-0.02144559	-0.05064393	-0.07093185
5/7/2012	-0.11459854	-0.06296296	-0.06587202	-0.02461140	-0.06979695	-0.02768805	-0.01136364	-0.02291476	-0.04041916
5/14/2012	-0.09398186	-0.06501976	-0.10678308	-0.07038513	-0.14461119	-0.07071666	-0.07553366	-0.06060038	-0.13416537
5/21/2012	0.00030331	0.01268231	-0.00751880	0.01857143	0.03987241	0.00715015	0.02664298	-0.03555023	0.02432432
5/29/2012	-0.04699818	-0.03746608	-0.03939394	-0.01683030	-0.05214724	-0.04006085	-0.02768166	-0.01615241	-0.06332454
6/4/2012	0.05504295	0.02060067	0.07728707	0.07703281	0.12944984	0.06920232	0.04181495	0.01999579	0.11361502
6/11/2012	0.04010857	0.01179220	0.04245974	0.04503311	0.12177650	-0.06966403	0.02391119	0.02434998	0.06492412
6/18/2012	0.02725428	-0.02120958	-0.01053371	0.00506971	-0.02681992	-0.00424854	-0.00333611	0.01007252	-0.00712589
6/25/2012	-0.00733841	0.02381463	0.03122782	0.03026482	-0.10761155	-0.02453333	-0.02008368	0.01017152	-0.18341308
7/2/2012	-0.04321865	-0.00408634	-0.03028218	-0.06364749	-0.07794118	-0.02624385	-0.05892400	0.01243830	-0.00292969
7/9/2012	0.06419019	0.02051552	-0.00638751	0.02091503	0.03189793	-0.02133633	-0.02994555	-0.00234009	-0.00293830
7/16/2012	-0.06031835	-0.03350516	-0.09071429	-0.09603073	-0.01236476	-0.02868617	-0.05051450	0.00000000	-0.03241650
7/23/2012	0.08826152	0.07946667	0.05970149	0.03399433	0.08137715	0.05434141	0.07980296	0.00664582	0.06598985
7/30/2012	-0.02157291	-0.00652174	0.02149741	0.01643836	-0.01447178	-0.03641457	-0.01733577	-0.00019418	0.01047619
8/6/2012	0.02428133	0.02019097	0.06023222	0.04177898	0.03377386	0.02209302	0.01207057	0.00369004	0.08576814
8/13/2012	0.00027248	0.00565468	-0.00136893	0.03363519	0.03551136	0.03811149	0.01284404	0.05572755	0.04774306
8/20/2012	0.00517570	0.01308774	-0.00205620	0.02002503	-0.02743484	0.05260274	0.01449275	-0.00623167	-0.01739851
8/27/2012	-0.00081301	0.01167464	0.03021978	-0.02085890	0.01410437	0.00260281	-0.00446429	0.03301365	-0.01939292
9/4/2012	0.05804177	0.10035944	0.13866667	0.10275689	0.09318498	0.09865005	0.10762332	0.02606677	0.13241617
9/10/2012	0.05793386	0.04323906	0.06791569	0.08522727	0.14631043	0.09215501	0.09149798	0.00556812	0.12452544
9/17/2012	-0.01671917	-0.03823336	-0.06359649	-0.04607330	-0.01109878	-0.00995240	-0.04154303	-0.00657553	-0.02633356
9/24/2012	-0.00985707	-0.02604524	-0.01990632	-0.03073546	-0.06621773	-0.07561189	-0.05727554	0.00000000	-0.03814147
10/1/2012	0.03807865	0.04952498	0.04540024	0.05549264	0.01682692	0.06713948	0.05090312	0.02229577	0.04542177
10/8/2012	-0.00215776	0.00745956	-0.01085714	-0.02145923	0.02127660	-0.00620292	-0.01484375	-0.01226785	0.01931035
10/15/2012	0.01681884	0.02845258	0.01270942	0.03508772	0.03472222	0.04725814	0.03489294	0.01621528	0.00879567

(ii) Find the linear regression of the JPM returns with respect to the returns of the other stocks. What is the approximation error of this linear regression?
Solution: The regression formula:

$$\begin{aligned}\text{JPM} \approx\ & 0.766295 \times \text{GS} - 0.077765 \times \text{MS} + 0.297951 \times \text{BAC} + 0.281170 \times \text{RBS} \\ & - 0.057614 \times \text{CS} - 0.414045 \times \text{UBS} + 0.147594 \times \text{RY} - 0.0008806 \times \text{BCS} - 0.003353\end{aligned}$$

the approximation error is $\sum_i e_i^2 = 0.01675761$.

(iii) Find the linear regression of the JPM returns with respect to the retusn of the other American financial companies, i.e. with respect to GS, MS, and BAC. What is the approximation error of this linear regression?
Solution: The regression formula:

$$\text{JPM} \approx 0.653710 \times \text{GS} - 0.037504 \times \text{MS} + 0.261285S \times \text{BAC} - 0.001394.$$

the approximation error is $\sum_i e_i^2 = 0.02246522$.

(iv) Find the linear regression of the JPM stock prices with respect to the prices of the other stocks. What is the approximation error of this linear regression? How does it compare with the approximation error of the linear regression of the JPM returns computed at (ii)?

Solution: The regression formula reads

$$\begin{aligned}\text{JPM} \approx\ & 0.0166 \times \text{GS} + 0.06682 \times \text{MS} + 1.47131 \times \text{BAC} + 1.20743 \times \text{RBS} \\ & 0.8990 \times \text{CS} - 2.53732 \times \text{UBS} + 0.35021 \times \text{RY} - 0.17046 \times \text{BCS} + 8.81655\end{aligned}$$

the approximation error is $\sum_i e_i^2 = 42.35296$. Compare to the error when regression with the return data, the approximation error is much larger in terms of absolute value.