АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ НАУЧНЫХ ИССЛЕДОВАНИЙ

S.A. Prokhorov, I.M. Kulikovskikh, D.V. Tselishev

SOLVING RECURRENCES FOR LAGUERRE FUNCTIONS TO PROVIDE SCIENTIFIC COMPUTING IN MOBILE TECHNOLOGIES

(Samara State Aerospace University, Samara, Russia)

Recently advances in mobile technology have improved computer processing power. To maximize the effects from these advances in hardware technology, the productivity of technology for developing concurrent networked middleware and application software should also increase. Although hardware advances, it is necessary to implement low-level software optimizations; the processing costs and effort to develop mobile software continues to increase [1, 2].

This article is motivated by the rapid development of mobile technologies. In spite of advantages of mobile technologies, such as portability, mobility, and productivity, the mobile application development determines the following minimal requirements: long response time, limited memory, and high processing costs. The main purpose of this research is to create lightweight algorithms according to these requirements. To discover the possibility of using mobile technologies for scientific computing, we developed the lightweight algorithms based on Fourier series expansion method. However, we offer to solve recurrences for orthogonal functions to keep to the requirements and to provide demanded accuracy of scientific computing [3].

We now state the problem. Let $L_k(\tau,\gamma)$ denote Laguerre functions [4] that are given by

$$kL_k(\tau,\gamma)=(2k-1-\tau)L_{k-1}(\tau,\gamma)-(k-1)L_{k-2}(\tau,\gamma)$$

in the Hilbert space $\tau\in R^+$ with the pole position $\gamma\in\Gamma$, where $\Gamma\in\{\gamma\in R:\gamma>0\}$.

With regard to $\int_0^\infty f^2(\tau)d\tau<\infty$, we can expressed a function $f(\tau)$ as an expanded series

$$f(\tau)=\sum_{k=0}^{m}\beta_k L_k(\tau,\gamma),$$

where the Laguerre coefficients can be defined as

$$\beta_k=\frac{\langle f(\tau),L_k(\tau,\gamma)\rangle}{\|L_k(\gamma)\|^2}, \qquad (1)$$

where $\|L_k(\gamma)\|^2$ is the norm of the Laguerre functions.

Then, we estimate the coefficients by numerical-analytical approach [5]

$$\beta_k = \frac{1}{\|L_k(\gamma)\|^2} \sum_{i=0}^{I_{\max}-1} \left(a_i \int_{t_i}^{t_{i+1}} L_k(t)dt + b_i \int_{t_i}^{t_{i+1}} tL_k(t)dt \right), \quad (2)$$

where a_i, b_i are linear interpolation coefficients in accordance with $f(\tau) = \sum_{i=0}^{I_{\max}-1} (a_i + b_i t)\delta_i$, δ_i – indicator function.

Using quadrature (2) considerably increases the accuracy of the research results in comparison with (1) if we define the integrals $\int L_k(\tau,\gamma)d\tau$ and $\int \tau L_k(\tau,\gamma)d\tau$ analytically by recurrence relations. To define these recurrences is the next problem.

Recurrences entail an enormous computational cost so it is very important to solve this problem in the best way. In accordance to types of recurrence [7] the orthogonal functions can be implemented as second order recurrences. There is common-sense rule for solving any recurrences [7]: to compute recurrence values do save all values in the array instead of do use a recursive program, because it takes exponential time. On the other hand, computing the Laguerre integrals leads to significant increase in memory size. To find the solution to this problem, we use the basic technique to solve recurrence that is called as telescoping [7].

Before we go any further, it is necessary to present $\int L_k(\tau,\gamma)d\tau$ and $\int \tau L_k(\tau,\gamma)d\tau$ using connection coefficients method [8]

$$\int L_k(\tau,\gamma)d\tau = \sum_{v=0}^{\infty} c_{k,v} L_v(\tau,\gamma) + C, \quad (3)$$

where

$$c_{k,v} = \left\langle L_v(\tau,\gamma), \int L_k(\tau,\gamma)d\tau \right\rangle = \begin{cases} 2, & if\ k = v; \\ 4(-1)^{k+v}, & if\ k < v; \\ 0, & otherwise \end{cases} \quad (4)$$

Substituting (4) in (3) gives

$$\int L_k(\tau,\gamma)d\tau = -4\sum_{v=0}^{\infty} (-1)^{k+v} L_v(\tau,\gamma) - 2L_k(\tau,\gamma) + C. \quad (5)$$

By analogy, we can define

$$\int \tau L_k(\tau,\gamma)d\tau = -(k+1)\int L_{k+1}(\tau,\gamma)dt + (2k+1)\int L_k(\tau,\gamma)dt - k\int L_{k-1}(\tau,\gamma)dt + C.$$

To support the theoretical results, we provided a series of computational experiments.

Here are the code snippets in Java in the case of direct using of the recurrences and solving the recurrences (5) to calculate the Laguerre integrals, respectively.

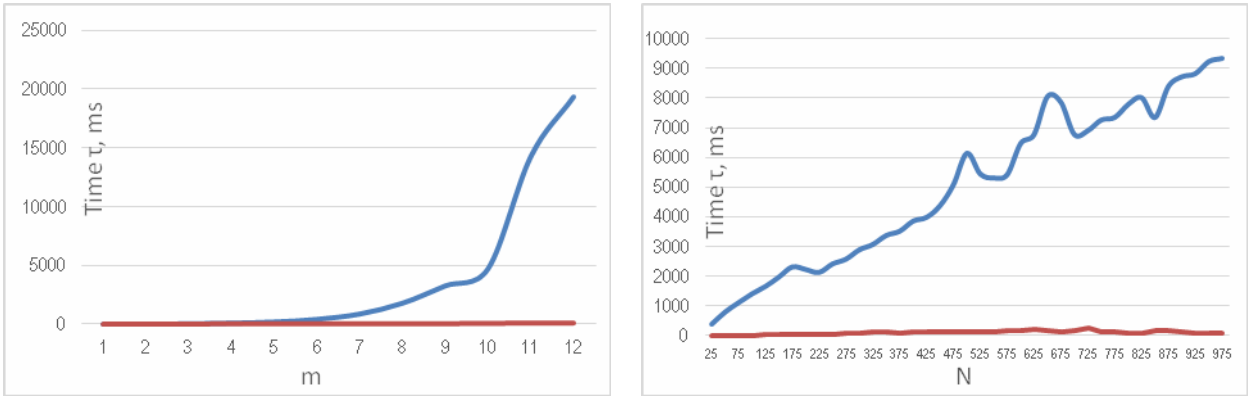
```
# direct using of the recurrences
public  static  double  recurrentFunctionLaggera
(double point, double gamma, int k) {
if (k == 0) {
     returnMath.exp(-gamma * point / 2);
 }
if (k == 1) {
```

```
returnMath.exp(-gamma * point / 2) * (1 - gamma *
point);
 }
return (2 * k - 1 - gamma * point) * recurrent-
FunctionLaggera (point, gamma, k - 1) / k - (k -
1) * recurrentFunctionLaggera (point, gamma, k -
2) / k;
 }
public static double recurrentIntegral1 (double
point, double gamma, int k) {
if (m == 0) {
return -2 * Math.exp(-gamma * point / 2) / gamma;
 }
doubletmp = 0;
tmp += -2 * recurrentFunctionLaggeraPoint(point,
gamma, k) / gamma;
for (inti = m - 1; i>= 0; i--) {
tmp += -2 * recurrentIntegral1(point, gamma, i);
 }
returntmp
 }
```

 # solving the recurrences

```
public static double basicIntegral1(double point,
double gamma, int k) {
double[] functionLaggera = basicFunctionLag-
gera(point, gamma, k);
double result = 0;
for (int v = 0; v <functionLaggera.length; v++) {
result += Math.pow(-1, k + v) * functionLag-
gera[v];
 }
result *= -4 / gamma;
result -= -2 * functionLag-
gera[functionLaggera.length - 1] / gamma;
return result;
 }
public static double[] basicFunctionLag-
gera(double point, double gamma, int k) {
double[] result = new double[k + 1];
result[0] = Math.exp(-gamma * point / 2);
if (k == 0) {
return result;
 }
result[1] = Math.exp(-gamma * point / 2) * (1 -
gamma * point);
for (inti = 2; i<result.length; i++) {
result[i] = (2 * i - 1 - gamma * point) * re-
sult[i - 1] / i - (i - 1) * result[i - 2] / i;
 }
return result;
 }
```

The results of the computational experiments to test these code snippets are depicted in Fig. 1 varying m (a) and number of intervals N (b) if $f(\tau) = \exp(\lambda\tau)(\cos\omega\tau + \lambda\sin\omega\tau/\omega)$ for given λ, ω. The experiments were provided by a tablet with the following features: 1GB RAM, quad-core CPU 1200 MHz, based on Android v. 4.4.1.



(a) (b)

Fig. 1. Results of the conducted experiments

From the data from Fig. 1 it is clear that solving recurrences gives enormous effect on providing scientific computing in mobile technologies.

References

1. Weir C., Noble J. Small Memory Software: Patterns for systems with limited memory, Addison Wesley, 2000.

2. Dougherty B., White J., Schmidt D.C. Model-driven auto-scaling of green cloud computing infrastructure, International Journal of Future Generation Computing Systems, Special Issue on Green Computing Systems, 28(2) (2012), 371 – 378.

3. Kulikovskikh I.M., Prokhorov S.A. Some lightweight algorithms for scientific computing in mobile technologies, Proceedings at the conference on Applied Mathematics and Scientific Computing, Croatia, Sibenik, 2013, pp. 40 – 41

4. Prokhorov S.A., Kulikovskikh I.M. Basic orthogonal functions and its applications. Part I. Orthogonal functions of exponential type, Samara Section of Russian Academy on Science, 2013.

5. Prokhorov S.A., Kulikovskikh I.M. Numerical-analytical approach to integrals calculation in the construction of orthogonal models,Vestn. Samar. Gos. Tech. Univ., Ser. Fizikomat. Nauki, 19 (2), 2009, pp. 140 – 146 [in russian]

6. Sedgewick R., Flajolet P. An introduction to the analysis of algorithms. Second edition, Pearson Education, 2013.

7. Doha E.H. On the connection coefficients and recurrence relations arising from expansions in series of Laguerre polynomials, J. Phys. A: Math. Gen. 36. 2003, pp. 5449 – 5462