УДК 519.6 + 517.983.54

О.В. Матысик, В.И. Басин

НЕЯВНАЯ ИТЕРАЦИОННАЯ ПРОЦЕДУРА И ЕЁ ПРИМЕНЕНИЕ ДЛЯ РЕШЕНИЯ МОДЕЛЬНОЙ НЕКОРРЕКТНОЙ ЗАДАЧИ В ГИЛЬБЕРТОВОМ ПРОСТРАНСТВЕ

В гильбертовом пространстве для решения операторного уравнения первого рода с положительным ограниченным и самосопряженным оператором изучается неявный итерационный метод. Для предложенного метода обосновано применение правила останова по невязке, что делает рассматриваемый итерационный метод эффективным и тогда, когда нет сведений об истокообразной представимости точного решения. Рассматриваемым методом решена численная модельная некорректная задача в виде уравнения Фредгольма первого рода.

1. Постановка задачи

В действительном гильбертовом пространстве H исследуется операторное уравнение I рода

$$Ax = y, \tag{1}$$

где A – положительный ограниченный и самосопряженный оператор, для которого нуль не является собственным значением, однако принадлежит спектру оператора A, и, следовательно, задача некорректна. Пусть $y \in R(A)$, т.е. при точной правой части y уравнение (1) имеет единственное решение x. Для отыскания этого решения применяется неявная итерационная процедура

$$(E + \alpha A)x_{n+1} = (E - \alpha A)x_n + 2\alpha y, \;\; x_0 = 0. \tag{2}$$

Предлагаемая схема является аналогом метода второго порядка $y_{m+1} - y_m = \frac{h}{2}(y'_{m+1} + y'_m)$ для обыкновенного дифференциального уравнения $y' = f(x, y)$ [1, с. 180].

В случае приближенной правой части y_δ, $\left(\|y - y_\delta\| \le \delta\right)$ соответствующие методу (2) итерации примут вид

$$(E + \alpha A)x_{n+1,\delta} = (E - \alpha A)x_{n,\delta} + 2\alpha y_\delta, \;\; x_{0,\delta} = 0. \tag{3}$$

Ниже, как обычно, под сходимостью метода (3) понимается утверждение о том, что приближения (3) сколь угодно близко подходят к точному решению уравнения при достаточно малых δ и $n\delta$ и достаточно больших n.

2. Сходимость метода в случае априорного выбора числа итераций

Методы (2)–(3) впервые были предложены в работе [2], в которой показана сходимость обоих методов (2) и (3) при условии $\alpha > 0$ и получена оценка погрешности в предположении, что решение является истокообразно представимым с некоторым показателем $s > 0$. В статье [2] доказана

Т е о р е м а 1. *Если* $x = A^s z$, $s > 0$, *и* $\alpha > 0$, *то справедлива оценка*

$$\|x - x_{n,\delta}\| \le s^s (2n\alpha e)^{-s} \|z\| + 2n\alpha\delta.$$

Отсюда при $n_{\text{опт}} = s\alpha^{-1}2^{-1}e^{-\frac{s}{s+1}}\delta^{-\frac{1}{s+1}} \| z \|^{\frac{1}{s+1}}$ получаем неравенство $\left\| x - x_{n,\delta} \right\|_{\text{опт}} \leq$ $\leq (1+s)e^{-\frac{s}{s+1}}\delta^{\frac{s}{s+1}} \| z \|^{\frac{1}{s+1}}$. Порядок последней оценки оптимален в классе задач с истокообразно представимыми решениями [3].

Очевидно, что оптимальная оценка погрешности не зависит от параметра α, но от него зависит $n_{\text{опт}}$. Так как на α нет ограничений сверху ($\alpha > 0$), то можно α выбрать так, чтобы $n_{\text{опт}} = 1$. Для этого достаточно взять $\alpha_{\text{опт}} = 2^{-1}se^{-s/(s+1)}\|z\|^{1/(s+1)}\delta^{-1/(s+1)}$. Таким образом, неявный метод (3) позволяет получить решение уже на первых шагах итераций.

3. Правило останова по невязке

Для решения уравнения (1) с ограниченным положительным и самосопряжённым оператором применяется метод итераций (3). Покажем возможность применения для этого метода правила останова по невязке [3–5].

Определим момент m останова итерационного процесса (3) условием

$$\left.\begin{array}{l} \left\| Ax_{n,\delta} - y_\delta \right\| > \varepsilon, (n < m) \\ \left\| Ax_{m,\delta} - y_\delta \right\| \leq \varepsilon, \end{array}\right\} \quad \varepsilon = b\delta, b > 1. \tag{4}$$

Предполагается, что при начальном приближении $x_{0,\delta}$ невязка достаточно велика, больше уровня останова ε, т. е. $\left\| Ax_{0,\delta} - y_\delta \right\| > \varepsilon$. Справедливы

Теорема 2. *Пусть* $A = A^* \geq 0, \|A\| \leq M$ *и пусть момент останова* $m = m(\delta)$ *в методе* (3) *выбирается по правилу* (4). *Тогда* $x_{m,\delta} \to x$ *при* $\delta \to 0$.

Теорема 3. *Пусть выполнены условия теоремы* 2 *и пусть* $x = A^s z, s > 0$. *Тогда справедливы оценки* $m \leq 1 + \dfrac{s+1}{4\alpha}\left[\dfrac{\|z\|}{(b-1)\delta}\right]^{1/(s+1)}$,

$$\left\| x_{m,\delta} - x \right\| \leq [(b+1)\delta]^{s/(s+1)}\|z\|^{1/(s+1)} + 2\alpha\left\{1 + \frac{s+1}{4\alpha}\left[\frac{\|z\|}{(b-1)\delta}\right]^{1/(s+1)}\right\}\delta. \tag{5}$$

Замечание 1. *Порядок оценки* (5) есть $O\left(\delta^{s/(s+1)}\right)$ *и, как следует из* [3], *он оптимален в классе задач с истокопредставимыми решениями.*

Замечание 2. *Используемое в формулировке теоремы* 3 *предположение порядка* $s > 0$ *истокопредставимости точного решения не потребуется на практике, так как оно не содержится в правиле останова* (4). *И тем не менее в теореме* 3 *утверждается, что будет автоматически выбрано количество итераций* m, *обеспечивающих оптимальный порядок погрешности. Но даже если истокопредставимость точного решения отсутствует, останов по невязке* (4), *как показывает теорема* 2, *обеспечивает сходимость метода, т.е. его регуляризующие свойства.*

4. Численный модельный пример

4.1 Формулировка и описание алгоритма решения модельной задачи

Задача. *Решаем в пространстве* $L_2(0,1)$ *модельную задачу в виде уравнения*

$$\int_0^1 K(t,s)\, x(s)\, ds = y(t), \quad 0 \le t \le 1 \tag{6}$$

с симметричным положительным ядром $K(t,s) = \begin{cases} t(1-s),\ 0 \le t \le s \le 1, \\ s(1-t),\ 0 \le s \le t \le 1, \end{cases}$ *точной правой частью* $y(t) = \dfrac{t(t-1)(t^2-t-1)}{12}$ *и точным решением* $x(t) = t(1-t).$

Обычно на практике мы не знаем точной функции $y(t)$, а вместо нее известны значения приближенной функции $\tilde{y}(t)$ в некотором числе точек с определенной, часто известной погрешностью δ, и по этим приближенным данным требуется приближенно найти решение. Чтобы имитировать эту ситуацию, будем считать заданными значения $\tilde{y}_i$, $i = \overline{1,m}$, полученные следующим образом: $\tilde{y}_i = [\, y(t_i) \cdot 10^k + 0{,}5] / 10^k$, где $y(t_i)$ – значения функции $y(t)$ в точках $t_i = ih$, $i = \overline{1,m}$, $h = 1/m$. Квадратные скобки означают целую часть числа и $k = 4$. При $k = 4$ величина погрешности $\delta = 10^{-4}$. Действительно, имеем

$$\int_0^1 [y(t) - \tilde{y}(t)]^2\, dt \approx \sum_{i=1}^{m} [y(t_i) - \tilde{y}_i]^2 h \le mh\left(10^{-k}\right)^2 = 10^{-2k}.$$

Заменим интеграл в уравнении (6) квадратурной суммой, например, по формуле правых прямоугольников с узлами $s_j = jh$, $j = \overline{1,m}$, $h = 1/m$, т. е. $\int_0^1 K(t,s)x(s)ds \approx$ $\approx \sum_{j=1}^{m} K(t,s_j)hx_j$. Тогда получим равенство $\sum_{j=1}^{m} K(t,s_j)hx_j + \rho_m(t) = y(t)$, где $\rho_m(t)$ – остаток квадратурной замены. Записав последнее равенство в точках $t_i = ih$, $i = \overline{1,m}$, получим уравнения $\sum_{j=1}^{m} K(t_i,s_j)hx_j + \rho_m(t_i) = y(t_i)$, $i = \overline{1,m}$. Отбросив теперь остаточный член, получим линейную алгебраическую систему уравнений относительно приближённого решения

$$\sum_{j=1}^{m} K(t_i,s_j)hx_j = \tilde{y}_i, \quad i = \overline{1,m}. \tag{7}$$

Выберем для определённости $m = 32$ и будем решать систему (7) методом итераций (3), который в дискретной форме запишется

$$x_i^{(n+1)} + \alpha \sum_{j=1}^{m} K(t_i,s_j)x_j^{(n+1)}h = x_i^{(n)} - \alpha \sum_{j=1}^{m} K(t_i,s_j)x_j^{(n)}h + 2\alpha\tilde{y}_i, i = \overline{1,m}. \tag{8}$$

При решении задачи итерационным методом (3) вычислялись: $\left\|Ax^{(n)}-\tilde{y}\right\|_m=\left\{\sum_{i=1}^{m}\left[\sum_{j=1}^{m}K(t_i,s_j)hx_j^{(n)}-\tilde{y}_i\right]^2 h\right\}^{1/2}$ – дискретная норма невязки, $\left\|x^{(n)}\right\|_m=\left\{\sum_{i=1}^{m}\left[x_i^{(n)}\right]^2 h\right\}^{1/2}$ – норма приближённого решения и дискретная норма разности между точным и приближённым решениями $\left\|x-x^{(n)}\right\|_m=\left\{\sum_{i=1}^{m}\left[x(t_i)-x_i^{(n)}\right]^2 h\right\}^{1/2}$.

Оператор, описанный выше интегральным уравнением, непрерывен, взаимнооднозначен и аддитивен. Задача была решена методом (3) при $\delta=10^{-4}$. Результаты счёта приведены в таблице 1 (ввиду симметрии приведена лишь половина таблицы). Для решения предложенной задачи сведений об истокопредставимости точного решения не потребовалось, так как здесь воспользовались правилом останова по невязке (4), выбрав уровень останова $\varepsilon=1{,}5\delta$. Пример счёта показал, что для достижения оптимальной точности методом итераций (3) при $\alpha=9$ требуется только одна итерация, что соответствует результатам раздела 2. На рисунке 1 изображены графики точного решения и приближённого решения, полученного методом (3) при $\delta=10^{-4}$.

4.2 Результат работы программы

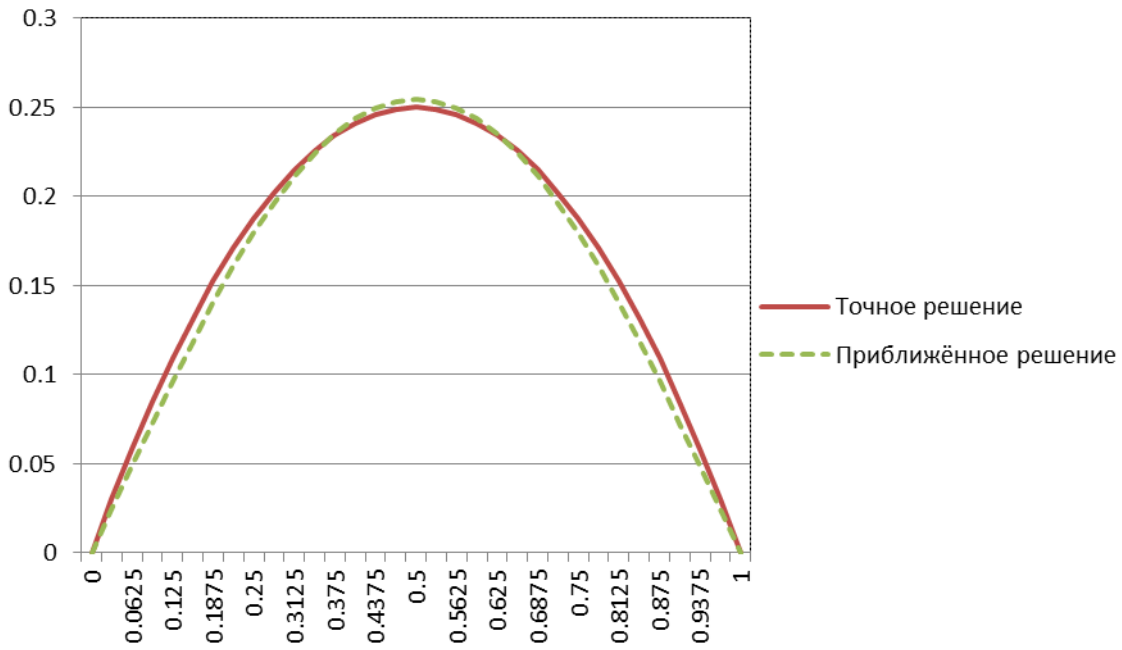

Рисунок 1

Таблица 1

Узлы t_i	Правые части $y(t_i)$	Точное решение $x(t_i)$	Приближённое решение, полученное методом (3)
			$\delta = 10^{-4}$
0	**0**	**0**	**0**
0.0312	**0.00259**	**0.03027**	**0.02429**
0.0625	**0.00517**	**0.05859**	**0.04865**
0.0937	**0.00768**	**0.08496**	**0.07275**
0.125	**0.01011**	**0.10938**	**0.09629**
0.1562	**0.01243**	**0.13184**	**0.11898**
0.1875	**0.01463**	**0.15234**	**0.14056**
0.2187	**0.01668**	**0.17089**	**0.1608**
0.2500	**0.01855**	**0.1875**	**0.17948**
0.2812	**0.02025**	**0.20215**	**0.19641**
0.3125	**0.02175**	**0.21484**	**0.21142**
0.3437	**0.02304**	**0.22559**	**0.22437**
0.375	**0.02411**	**0.23438**	**0.23514**
0.4062	**0.02495**	**0.24121**	**0.24361**
0.4375	**0.02555**	**0.24609**	**0.24972**
0.4687	**0.02591**	**0.24902**	**0.25341**
0.5000	**0.02604**	**0.25000**	**0.25464**
$\left\|Ax^{(n)} - \tilde{y}\right\|_m$			**0.00015**
$\left\|x^{(n)}\right\|_m$			**0.17972**
$\left\|x - x^{(n)}\right\|_m$			**0.00798**

4.3 Исходный код программы (на языке C#)

Вспомогательные классы Matrix и Vector:

```
using System;
using MTask.CoreClass;
namespace Helpers.Classes
{
   public class Matrix
   {
      private int _height, _width;
      private double[,] data;
      private double[] t;
      public int Height { get { return _height; } }
      public int Width { get { return _width; } }
      public double[,] Data { get { return data; } set { data = value; } }
      public Matrix(int height, int width)
      {
         _height = height; _width = width;
         data = new double[_height, _width];
      }
      private int getFirstNonZero(int RowNum)
      {
```

```
    for (int i = 0; i < _width; i++)
      if (Data[RowNum, i] != 0) return i;
    return _width;
  }
  private void Sort(Vector temp)
  {
    t = new double[_width];
    for (int i = 0; i < _height - 1; i++)
      for (int j = i + 1; j < _height; j++)
        if (getFirstNonZero(i) > getFirstNonZero(j))
        {
          for (int h = 0; h < _width; h++) t[h] = Data[i, h];
          for (int h = 0; h < _width; h++) Data[i, h] = Data[j, h];
          for (int h = 0; h < _width; h++) Data[j, h] = t[h];
          double t1 = temp.Data[i];
          temp.Data[i] = temp.Data[j];
          temp.Data[j] = t1;
        }
  }
  private void ToStage(Vector temp)
  {
    double alpha;
    for (int i = 0; i < _height; i++)
    {
      if (Data[i, i] == 0) continue;
      for (int j = i + 1; j < _height; j++)
      {
        if (Data[j, i] == 0) continue;
        alpha = Data[j, i] / Data[i, i];
        for (int h = 0; h < _width; h++)
          Data[j, h] -= Data[i, h] * alpha;
        temp.Data[j] -= temp.Data[i] * alpha;
        Data[j, i] = 0;
      }
      Sort(temp);
    }
    Sort(temp);
  }
  public void CopyFrom(Matrix m)
  {
    Data = new double[m._height, m._width];
    _height = m._height; _width = m._width;
    for (int i = 0; i < m._height; i++)
      for (int j = 0; j < m._width; j++)
        Data[i, j] = m.Data[i, j];
  }
  public Vector GetSolution(Vector b)
  {
    Vector res = new Vector();
    Vector temp = new Vector();
    temp.CopyFrom(b);
    ToStage(temp);
    if (Data[_height - 1, _width - 1] == 0)
      res.Data[_width - 1] = 0;
```

```
            else
               res.Data[_width - 1] = temp.Data[_width - 1] /
                                 Data[_height - 1, _width - 1];
            for (int i = _height - 2; i >= 0; i--)
            {
               if (Data[i, i] == 0)
                  res.Data[i] = 0; continue;
               double sum = 0;
               for (int j = i + 1; j < _width; j++)
                  sum += Data[i, j] * res.Data[j];
               temp.Data[i] -= sum;
               res.Data[i] = temp.Data[i] / Data[i, i];
            }
            return res;
         }
      }
      public class Vector
      {
         private int _dimension;
         private double[] data;
         public int Dimension { get { return _dimension; } }
         public double[] Data { get { return data; } set { data = value; } }
         public Vector(int dimension)
         {
            _dimension = dimension;
            data = new double[dimension];
         }
         public Vector()
         {
            _dimension = Core.M;
            data = new double[Core.M];
         }
         public void CopyFrom(Vector w)
         {
            this._dimension = w._dimension;
            for (int i = 0; i < _dimension; i++)
               this.Data[i] = w.Data[i];
         }
      }
}
```

<u>Вспомогательный класс Core</u> (класс, решающий поставленную задачу):

```
using System;
using Helpers.Classes;
namespace MTask.CoreClass
{
   public class Core
   {
      private double h;
      private Matrix K, Z;
      private double n1, n2, n3;
      private int iterations;
      private Vector solution, v1, v2, y, x0;
      public const int M = 32, k = 4;
      public const double alpha = 9, a = 0, b = 1, delta = 1e-4, eps = 1.5 * delta;
```

```
public double Norm1 { get { return n1; } }
public double Norm2 { get { return n2; } }
public double Norm3 { get { return n3; } }
public Vector PreciselySolution { get { return v2; } }
public Vector Solution { get { return solution; } }
public int TotalIterations { get { return iterations; } }
public Core()
{
   v1 = new Vector(); v2 = new Vector(); x0 = new Vector();
   y = new Vector(); solution = new Vector();
   K = new Matrix(M, M);
}
private void GetMatrix()
{
   for (int i = 0; i < M; i++) v1.Data[i] = i * h;
   for (int i = 0; i < M; i++)
      for (int j = 0; j < M; j++)
         K.Data[i, j] = GetK(v1.Data[i], v1.Data[j]);
}
private double GetPoint(int i)
{
   return i * h;
}
public double GetY(int i)
{
   double f = GetPoint(i);
   return f * (f - 1) * (f * f - f - 1) / 12;
}
private double GetK(double t, double s)
{
   if (t <= s) return t * (1 - s);
   if (s <= t) return s * (1 - t);
   return 0;
}
private Vector GetRightPart(Vector xn)
{
   Vector res = new Vector();
   for (int k = 0; k < M; k++)
   {
      double t1 = GetY(k);
      double t2 = 0;
      for (int j = 0; j < M; j++) t2 += K.Data[k, j] * xn.Data[j] * h;
      res.Data[k] = xn.Data[k] - alpha * t2 + 2 * alpha * t1;
   }
   return res;
}
public double GetQ(Vector v)
{
   double sum = 0;
   for (int i = 0; i < M; i++)
   {
      double t = 0;
      for (int j = 0; j < M; j++)
         t += K.Data[i, j] * h * v.Data[j];
```

```
            t -= (GetY(i) * Math.Pow(10, k) + 0.5) / Math.Pow(10, k);
            t *= t;
            t *= h;
            sum += t;
         }
         return Math.Sqrt(sum);
      }
      public void Solve()
      {
         h = (b - a) / M;
         iterations = 0;
         x0 = new Vector();
         solution = new Vector();
         GetMatrix();
         Z = new Matrix(M, M);
         for (int i = 0; i < M; i++)
         {
            double t0 = 1 + alpha * K.Data[i, 0] * h;
            Z.Data[i, i] = t0;
            for (int j = 0; j < M; j++)
            {
               if (i == j) continue;
               Z.Data[i, j] = alpha * K.Data[i, j] * h;
            }
         }
         Vector bb = GetRightPart(x0);
         solution = Z.GetSolution(bb);
         n1 = GetQ(solution);
         for (int i = 0; i < M; i++)
            v2.Data[i] = GetPoint(i) * (1 - GetPoint(i));
         double t = 0;
         for (int i = 0; i < M; i++)
            t += solution.Data[i] * solution.Data[i] * h;
         n2 = Math.Sqrt(t); t = 0;
         for (int i = 0; i < M; i++)
            t += (v2.Data[i] - solution.Data[i]) *
                 (v2.Data[i] - solution.Data[i]) * h;
         n3 = Math.Sqrt(t);
      }
   }
}
```

Главный класс программы:

```
using System;
using MTask.CoreClass;
namespace MTask
{
   class Program
   {
      static void Main(string[] args)
      {
         Core core = new Core();
         core.Solve();
         Console.WriteLine("Precisely solution: ");
         for (int i = 0; i < 32; i++)
```

```
            Console.WriteLine("x" + i + ": " + core.PreciselySolution.Data[i]);
        Console.WriteLine();
        Console.WriteLine("Approximately solution: ");
        for (int i = 0; i < 32; i++)
            Console.WriteLine("x" + i + ": " + core.Solution.Data[i]);
        Console.WriteLine();
        Console.WriteLine("Total iterations: ");
        Console.WriteLine(core.TotalIterations);
        Console.WriteLine();
        Console.WriteLine("Norm: ");
        Console.WriteLine(core.Norm1);
        Console.WriteLine(core.Norm2);
        Console.WriteLine(core.Norm3);
        Console.WriteLine();
        Console.WriteLine("Ti");
        for (int i = 0; i < 17; i++)
            Console.WriteLine((Core.b - Core.a) / Core.M * (i));
        Console.WriteLine();
        Console.WriteLine("Yi");
        for (int i = 0; i < 17; i++)
            Console.WriteLine(core.GetY(i));
    }
  }
}
```

СПИСОК ЛИТЕРАТУРЫ

1. Бабушка, Н. Численные процессы решения дифференциальных уравнений / Н. Бабушка, Э. Витасик, М. Прагер. – М. : Наука, 1969. – 368 с.

2. Лисковец, О.А. Об одном итеративном методе решения уравнений I-го рода / О.А. Лисковец, В.Ф. Савчук // Вопросы прикл. математики: сб. ст. / СО АН СССР. – Иркутск, 1975. – С. 31–35.

3. Вайникко, Г.М. Итерационные процедуры в некорректных задачах / Г.М. Вайникко, А.Ю. Веретенников. – М. : Наука, 1986. – 178 с.

4. Емелин, И.В. К теории некорректных задач / И.В. Емелин, М.А. Красносельский // Доклады АН СССР. – 1979. – Т.244, № 4. – С. 805–808.

5. Матысик, О.В. О регуляризации операторных уравнений в гильбертовом пространстве / О.В. Матысик // Доклады НАН Беларуси. – 2005. – Т. 49, № 3. – С. 38–43.

O.V. Matysik, V.I. Basin. **Implicite Iteration Procedure and its Application for Decision the Model Incorrect Problems in the Hilbert Space**

In the Hilbert space for solving operator equations of type I with affirmative limited and self–conjugate operator the implicite iteration method is studied. The application of a rule residual stop for the offered method has been proved, which makes viewed iteration method quite effective even when there are no data about source representability of exact solution. The viewed method solves numerical model incorrect task as equation Fredgolm's of one sort.

Рукапіс паступіў у рэдкалегію 20.09.2011 г.