Структура данных для действий над профилями Стокса

Основной результат моделирования процессов в солнечной фотосфере – это профили Стокса фраунгоферовых линий. Чаще всего их требуется вывести в виде, пригодном для сравнения с наблюдаемыми профилями. Однако ряд задач требует дальнейших действий.
Во-первых, если мы перестаем считать наблюдаемую область Солнца однородной, то мы получаем набор профилей – различных для каждого элементарного однородного участка. Этот набор профилей мы должны микшировать в один, чтобы сравнивать его с наблюдательным материалом. Такая задача встает, например,

- при анализе потока излучения от Солнца как звезды,
- при анализе спектра спокойного Солнца с учетом неразрешенных ячеек грануляции,
- при анализе спектров тени, которые всегда включают в себя долю рассеянного света полутени и фотосферы.

Во-вторых, профили могут нас интересовать в момент прохождения излучения через слои фотосферы – если мы хотим в каждом шаге по элементарным слоям учитывать перераспределение по частотам. Если учитывать также перераспределение и по направлениям, то надо иметь набор профилей для разных μ.

Векторы UI,UQ,UU,UV.

Профиль Стокса – это набор значений интенсивности поляризованного света для соответствующих точек в шкале длин волн в окрестностях спектральной линии. Назовем профили параметров I,Q,U,V векторами (т.е. одномерными массивами) Унно – по имени автора, первым нашедшего решение для профилей Стокса [1]. Для разных задач нужно разное число точек на профиле. Поэтому для хранения векторов Унно в программе используем динамические массивы `UI,UQ,UU,UV` (мы применяем структуры данных Delphi, т.к. наша программа SunWorld сделана в этой системе).

Вектор UL.

С четырьмя векторами Унно связан соответствующий вектор длин волн, который хранится в динамическом массиве `UL`. Заметим, что счет точек, начиная с нуля не очень удобен для восприятия человеком. Поэтому в динамических массивах, которые обязательно начинаются с нуля, нулевую точку просто игнорируем.

Размерность массивов NDL.

Для векторов задана их размерность – число точек, которое мы будем хранить в целой переменной `NDL`.

Параметр CONT.

Привычная запись профилей Стокса, это та, в которой интенсивности нормированы, т.е. поделены на уровень интенсивности в непрерывном спектре. Интенсивность континуума будем хранить в вещественной переменной `CONT`.

Первое приближение структуры TUNNO.

Таким образом, как первое приближение структуры данных имеем:

```
type

  TARe : array of real;    (* динамический массив *)

TUNNO = record
  NDL  : integer; (* число длин волн              *)
  CONT : real;    (* уровень континуума           *)
  UL   : TARe;    (* вектор  длин волн            *)
  UI   : TARe;    (* профиль интенсивности        *)
  UQ   : TARe;    (* профиль линейной поляризации *)
  UU   : TARe;
  UV   : TARe;    (* профиль круговой поляризации *)
end;
```

TUNNO не просто запись, над её данными будем производить действия, поэтому преобразуем её в «самодвижущуюся» запись, т.е. в объект, точнее в класс Delphi.

```
TUNNO = class (TObject)
  ...
```

Флаг KL.

Разные задачи требуют разные типы вектора длин волн. Ряд задач, где нужно работать со многими профилями требуют одного и того же вектора длин волн. Переменные, описывающие динамические массивы являются ссылками. Поэтому можно определить, заполнить один массив UL в одном объекте TUNNO, а во всех прочих объектах копировать ссылку. Добавим в структуру флаг KL со следующими значениями, которые указывают, что структура содержит:

```
0 – и свой вектор длин волн и векторы Унно
1 – только вектор длин волн, векторы Унно пустые
2 – векторы Унно и ссылку на вектор длин волн
```

В программе определено в общем случае много объектов типа TUNNO, какие-то из них могут быть специальными объектами, описывающими только вектор длин волн. Для таких объектов KL = 1.

Флаг KSym.

Если расчет проводится для одной линии и поле лучевых скоростей не задано, то векторы Унно оказываются симметричными, и расчет достаточно проводить только для половины линии. В противном случае надо считать полный профиль. Чтобы различать эти два случая введем флаг KSym:

```
1 – симметричный профиль (половина профиля)
2 – полный профиль
```

Флаг KDL.

Для расчетов единичных линий удобно указывать для каждой точки расстояние от центра линии в миллиангстремах. Для расчетов спектрального диапазона логичнее указывать непосредственно длины волн в ангстремах. Тип массива длин волн определяет флаг KDL:

```
1 – абсолютный, в ангстремах
0 – относительный, в миллиангстремах
```

Параметр LAMB.

Если тип массива длин волн относительный, необходимо бывает рассчитать собственно длину волны, для этого вводим поле LAMB – опорная длина волны.

Индекс ILAM.

Для сложных случаев опорная длина волны может приходиться не на первую точку профиля, а на произвольную, введем её индекс ILAM

Фрагмент структуры TUNNO, описывающей длины волн.

Итак, дополняем структуру TUNNO:

```
TUNNO = class (TObject)
  . . .
  (*-------------------------------------------------------------------*)
  (*              параметры, описывающие массив длин волн              *)
  (*-------------------------------------------------------------------*)

  NDL  : integer;(* число точек профиля                                 *)
  KL   : integer;(* флаг - тип профиля в отношении длин волн          *)
                 (* 0 - и свой вектор длин волн и векторы Унно        *)
                 (* 1 - только вектор длин волн, векторы Унно пустые  *)
                 (* 2 - векторы Унно и ссылка на вектор длин волн     *)
  KSym : integer;(* флаг симметрии                                    *)
                 (* 1 - симметричный профиль (половина профиля)       *)
                 (* 2 - полный профиль                                *)
  KDL  : integer;(* флаг - тип массива длин волн                      *)
                 (* 0 - относительный, в миллиангстремах              *)
                 (* 1 - абсолютный, в ангстремах                      *)
  LAMB : real;   (* опорная длина волны                               *)
  ILAM : integer;(* индекс массивов, к которому привязана LAMB        *)
  . . .
  UL   : TARe;   (* вектор  длин волн                                 *)
  . . .
```

Флаг K4.

Если мы имеем дело с формированием линий в отсутствие магнитного поля, векторы UQ,UU,UV оказываются пустыми. На этот случай добавим флаг K4

```
0 – структура содержит все 4 вектора Унно
1 – в структуре определен только вектор UI
```

Параметры LGT и MJU.

Для наблюдения за процессом формирования профилей Стокса в глубине фотосферы для объектов TUNNO может потребоваться определение параметра текущей оптической глубины, которую будем задавать в виде log10(Tau) и хранить в переменной LGT и параметра, определяющего угол приходящего излучения, которую зададим как $\mu = \cos\theta$ - косинус угла между приходящим излучением и лучем зрения и будем хранить в переменной MJU.

Флаги KCONT, KLGT, KMJU.

Значения UI,UQ,UU,UV могут быть нормированы (поделены на величину CONT) или ненормированы. Определим для нормированных профилей KCONT = 1.

Значения LGT и MJU могут быть определены и неопределенны. Если значения определены, то соответственно KLGT и KMJU равны 1.

Подструктура для интерполяции.

Некоторые задачи требуют выдавать значение параметров Стокса I,Q,U,V в точках, промежуточных к длинам волн, заданным в векторе UL. В этом случае выходные значения нужно интерполировать. Мы используем интерполяцию сплайнами, как вариант существенно более точный, чем линейная интерполяция. Для интерполяции сплайнами требуется заполнять рабочие массивы. Поскольку они тоже динамические, то достаточно ненакладно их также добавить к описанию типа TUNNO и мы добавляем

```
(*-------------------------------------------------------------------*)
(*              массивы для интерполяции сплайнами                   *)
(*-------------------------------------------------------------------*)

KSPLINE  : integer; (* флаг=1 - массивы для интерполяции заполнены *)

SPLINEI  : TSPLINE;
SPLINEQ  : TSPLINE;
SPLINEU  : TSPLINE;
SPLINEV  : TSPLINE;

. . .
```

Хранение множества структур TUNNO в списке TLUNNO.

Объекты TUNNO можно определять прямо в вычислительных процедурах. Более общим подходом будет создавать и хранить их в общем списке

```
TLUNNO = class(TList)  (* хранитель всех профилей Стокса *)

end;   ,
```

а в вычислительные процедуры передавать по ссылке. Таким образом мы сможем распределять и освобождать память в одном месте и передавать профили – объекты TUNNO из одной процедуры в другую.

Окончательный вид структуры данных TUNNO.

Итак, у нас получился следующий окончательный вид:

```
TUNNO = class(TObject)
  (*-------------------------------------------------------------------*)
  (*              параметры, описывающие массив длин волн              *)
  (*-------------------------------------------------------------------*)

  NDL  : integer;(* число точек профиля                                 *)
  KL   : integer;(* флаг - тип профиля в отношении длин волн          *)
                 (* 0 - и свой вектор длин волн и векторы Унно        *)
                 (* 1 - только вектор длин волн, векторы Унно пустые  *)
                 (* 2 - векторы Унно и ссылка на вектор длин волн     *)
  KSym : integer;(* флаг симметрии                                    *)
                 (* 1 - симметричный профиль (половина профиля)       *)
                 (* 2 - полный профиль                                *)
  KDL  : integer;(* флаг - тип массива длин волн                      *)
                 (* 0 - относительный, в миллиангстремах              *)
                 (* 1 - абсолютный,    в ангстремах                   *)
  LAMB : real;   (* опорная длина волны                               *)
  ILAM : integer;(* индекс массивов, к которому привязана LAMB        *)


  (*-------------------------------------------------------------------*)
  (*              физические параметры и опции                         *)
  (*-------------------------------------------------------------------*)

  CONT : real;   (* уровень континуума                                *)
  LGT  : real;   (* значение log10(Tau), если оно определено          *)
  MJU  : real;   (* значение cos(Theta), если оно определено          *)

  KCONT: integer;(* 1 = значения UI,UQ,UU,UV поделены на CONT         *)
  KLGT : integer;(* 1 = значение LGT определено                       *)
  KMJU : integer;(* 1 = значение MJU определено                       *)

  K4   : integer;(* флаг учета поляризации излучения                 *)
                 (* 0 или 4 - заполнены все 4 вектора Унно           *)
                 (* 1       - заполнен только профиль интенсивности  *)

  (*-------------------------------------------------------------------*)
  (*              вектор длин волн и векторы Унно                      *)
  (*-------------------------------------------------------------------*)

  UL   : TARe;
  UI   : TARe;
  UQ   : TARe;
  UU   : TARe;
  UV   : TARe;

  (*-------------------------------------------------------------------*)
  (*             структуры для интерполяции сплайнами                  *)
  (*-------------------------------------------------------------------*)
  KSPLINE  : integer; (* флаг = 1 - структуры заполнены                *)

  SPLINEI  : TSPLINE;
  SPLINEQ  : TSPLINE;
  SPLINEU  : TSPLINE;
  SPLINEV  : TSPLINE;

(*=====================================================================*)
(*                процедуры и функции TUNNO                            *)
(*=====================================================================*)
```