# Field Statistics Module: Requirements and Design

MPAS Development Team

February 26, 2013

# Contents

# Chapter 1

# Summary

This document describes a module which can compute generic statistics given a field. The statistics described in this document are specifically time averages, and moments, however this could be extended at a later time.

# Chapter 2

# Requirements

## 2.1 Requirement: Field Time Average Module

Date last modified: 02/26/13
Contributors: (Doug Jacobsen)

Some cores require the ability to time average fields. It would be useful to have this capability built into the shared framework, so fields can arbitrarily be time averaged and written to an output stream.

## 2.2 Requirement: Moment Computation Module

Date last modified: 02/26/13
Contributors: (Doug Jacobsen)

In addition to time averages, cores might benefit from computing the moments of fields.

## 2.3 Requirement: Run-time configuration

Date last modified: 02/26/13
Contributors: (Doug Jacobsen)

The decision to compute time averages or moments of fields should be handled at run-time rather than build time.

# Chapter 3

# Design and Implementation

## 3.1 Implementation: Time Average Module

Date last modified: 02/26/13
Contributors: (Doug Jacobsen)

One issue with time averaging fields is they generally require the addition of another field, which represents the time average of the full field. Currently the structure for adding a time average field would include creating a new field, and within a core averaging the full field into that new field every time step.

An alternative, is to add another array to each field type as follows:

```
! Derived type for storing fields
type field5DReal

   ! Back-pointer to the containing block
   type (block_type), pointer :: block

   ! Raw array holding field data on this block
   real (kind=RKIND), dimension(:,:,:,:,:), pointer :: array
   real (kind=RKIND), dimension(:,:,:,:,:), pointer ::  tavgArray

   ! Information used by the I/O layer
   type (io_info), pointer :: ioinfo ! to be removed later
   character (len=StrKIND) :: fieldName
   character (len=StrKIND), dimension(:), pointer :: constituentNames => null()
   character (len=StrKIND), dimension(5) :: dimNames
   integer, dimension(5) :: dimSizes
   logical :: hasTimeDimension
   logical :: isSuperArray
   type (att_list_type), pointer :: attList => null()

   ! Pointers to the prev and next blocks for this field on this task
   type (field5DReal), pointer :: prev, next

   ! Halo communication lists
   type (mpas_multihalo_exchange_list), pointer :: sendList
   type (mpas_multihalo_exchange_list), pointer :: recvList
   type (mpas_multihalo_exchange_list), pointer :: copyList
end type field5DReal
```

At runtime, the namelist for a particular field will determine if this field is supposed to be time averaged or not. If the field is determined to be time averaged, this new tavgArray array will be allocated.

A new time average module will be created, which will include checks on all fields that have 2 or more time levels defined and will compute the time averages of all fields that have an allocated tavgArray array.

This new module will largely be created by Registry, and will allow cores to call a single set of subroutines to zero, increment, and normalize time averaged fields.

One requirement for this formulation, is that a field and it's time averages share the same output streams. This means a stream can't contain only the time average of a field, or just the field. If one is to be written out, and both are computed, both are written out.

## 3.2 Implementation: Moment computation module

Date last modified: 02/26/13
Contributors: (Doug Jacobsen)

The implementation of a moment computation module largely follows the time average module formulation. A moment array is added to each field, along with a moment ID array, as follows.

```
! Derived type for storing fields
type field5DReal

   ! Back-pointer to the containing block
   type (block_type), pointer :: block

   ! Raw array holding field data on this block
   real (kind=RKIND), dimension(:,:,:,:,:), pointer :: array
   real (kind=RKIND), dimension(:,:,:,:,:,:), pointer ::  tavgMoments
   real (kind=RKIND), dimension(:,:,:,:,:,:), pointer ::  momentsArray
   real (kind=RKIND), dimension(:), pointer ::  momentIDs

   ! Information used by the I/O layer
   type (io_info), pointer :: ioinfo ! to be removed later
   character (len=StrKIND) :: fieldName
   character (len=StrKIND), dimension(:), pointer :: constituentNames => null()
   character (len=StrKIND), dimension(5) :: dimNames
   integer, dimension(5) :: dimSizes
   logical :: hasTimeDimension
   logical :: isSuperArray
   type (att_list_type), pointer :: attList => null()

   ! Pointers to the prev and next blocks for this field on this task
   type (field5DReal), pointer :: prev, next

   ! Halo communication lists
   type (mpas_multihalo_exchange_list), pointer :: sendList
   type (mpas_multihalo_exchange_list), pointer :: recvList
```

```
  type (mpas_multihalo_exchange_list), pointer :: copyList
end type field5DReal
```

The momentsArray array is one higher dimension than the field array, to allow multiple moments to be stored within the same array. Moments to compute will be specified in the I/O namelist, the same way time averaging is specified. Potentially a space delimited list of moment numbers to compute will be specified as follows:

```
&fieldName
        config_fieldName_moments = "2 4"
/
```

This input string will be broken apart and stored in the momentIDs array. A call to a shared subroutine for moment computation will compute all requested moments, and store them in the momentsArray array. As with the time averages, these arrays are not allocated if moments are not requested for the specific field. When the moment computation module is called, all fields are checked to see if any moments should be computed, based on if the momentIDs array is allocated with values or not.

Additionally, a tavgMoments array is provided to compute time averages of the moments.

As with the time averages, all moments and time averages of moments are share the same output streams as the parent field.

## 3.3 Implementation: Run-time configuration

Date last modified: 02/26/13
Contributors: (Doug Jacobsen)

The implementation for run-time configuration of field statistics requires the implementation of a run-time I/O layer. After the run-time I/O layer is complete, the namelist that controls I/O will contain flags for each field to determine which statistics (if any) should be computed.

At run-time, these flags will be checked and MPAS will either compute the statistics for a particular field or will skip the field all together.

# Chapter 4

# Testing

## 4.1 Testing and Validation: Run-time I/O Layer

Date last modified: 02/26/13
Contributors: (Doug Jacobsen)