Joint Hurricane Testbed: Year-2 Final Report

Dynamic Initialization to Improve Tropical Cyclone Intensity and Structure Forecasts

Principle Investigator: Chi-Sann Liou
Naval Research Laboratory, Monterey, CA

June 2007

1. Introduction

Development of a proper initialization method for tropical cyclone modeling remains a challenging task for tropical cyclone modelers. The difficulty stems from both inadequate observational data coverage and inadequate data analysis methodology. For most operational centers, the data analysis for tropical cyclone initialization is performed by 3-dimensional variational (3D-Var) analysis with poor quality background fields and a geostrophic-wind balance constraint. As a consequence, initial conditions provided to dynamic models are usually unbalanced with large oscillations in early forecast periods (Fig.1). The large initial oscillations prohibit forecasters from using the potentially most accurate intensity and structure tendency forecasts at early periods for forecasting guidance. The goal of this project is to develop and implement a dynamic initialization procedure to balance initial conditions for tropical cyclone forecasts by the hurricane Weather Research Forecasting (HWRF) model and the Coupled Ocean-Atmosphere Mesoscale Prediction System (COAMPS®[1]).

There are two types of initialization procedure used in numerical weather prediction to remove unbalanced components from initial conditions: static initialization and dynamic initialization. Static initialization removes the tendency of high frequency components from initial conditions by solving the nonlinear balance equation iteratively for each high frequency mode. Dynamic initialization removes the tendency of high frequency components from initial conditions by damping or filtering out high frequency solutions through back and forth integrations. The cost of static initialization is little, but it may not be suitable for tropical cyclone initialization since it can only include adiabatic processes. The iteration may not converge when diabatic terms are included in the balance calculation (Williamson and Temperton 1981, Rasch 1985). On the other hand, the cost of dynamic initialization is higher than that for static initialization, but it can include all diabatic processes in searching for balanced initial conditions. Digital filter initialization is one kind of dynamic initialization that filters out high frequency components through use of a discrete Fourier Transform. Lynch and Huang (1992) have analytically shown that adiabatic digital filtering is identical to Machenhauer's criterion for static initialization when the nonlinear terms are assumed constant in time. They have further shown that with slowly varying nonlinear forcing adiabatic digital filtering can still precisely remove high-frequency components from the initial conditions, while some high-frequency components will remain in the initial conditions after static initialization. Huang and Lynch (1993) have demonstrated that diabatic digital filtering is superior to the corresponding adiabatic digital filtering in reducing initial forecast noise and providing a better organized initial pressure-tendency field. The digital filter procedure, especially diabatic digital filtering, takes the time variation of nonlinear terms and the diabatic forcing into account so that the

[1] COAMPS® is a trademark of Naval Research Laboratory

initialization should lead to a better balance condition. Including diabatic processes in the initialization procedure is particularly important for tropical cyclone initialization since the diabatic forcing dominates the flow balance near a tropical cyclone core. The diabatic digital filter (DDF) initialization is the procedure we have developed and implemented for tropical cyclone forecasts by HWRF and COAMPS.

2. DDF Initialization

All initialization procedures for numerical modeling assume that unbalanced components are in high frequencies and they can be removed by filtering out tendency components with frequencies higher than a cutoff frequency. A traditional dynamic initialization procedure uses a selective damping mechanism during back and forth pre-forecast integrations to remove high-frequency components from initial conditions. On the other hand, the digital filter initialization uses a discrete, inverse Fourier transform of a lowpass filter to very selectively filter out high-frequency oscillations from initial conditions. In a frequency domain, the digital filtering operates as

$$F_{out}(\varpi) = F_{in}(\varpi) * H(\varpi),\ \ where\ H(\varpi) = 1,\ \ |\varpi| \le \varpi_c \qquad (1)$$
$$= 0,\ \ |\varpi| > \varpi_c$$

where ϖ_c is the cutoff frequency, and F_{in} and F_{out} are the frequency component before and after the filtering, respectively. In a physical domain, after applying the convolution theory and truncated inverse Fourier transform, the digital filtering acts as

$$\tilde{f}_n = \sum_{k=-\infty}^{\infty} h_k f_{n-k} \approx \sum_{k=-N}^{N} h_k f_{n-k} \qquad (2)$$

where $\tilde{f}_n$ is the filtered field at time n, f_{n-k} is the original field at time (n-k), $h_k = \frac{\sin(k\varpi_c \Delta t)}{k\pi}$ is the inverse Fourier transform of $H(\varpi)$ with Δt time step, and $2N * \Delta t$ is equal to the cutoff period. To improve the convergence rate and reduce negative ripples, a window-function w_k is usually applied together with the filtering weight h_k as

$$\tilde{f}_n = \sum_{k=-N}^{N} h_k w_k f_{n-k} \qquad (3)$$

After a broad survey and careful evaluation, we have chosen the Dolph-Chebyshev window which gives the window function as

$$w_n = \frac{1}{w_0(2N+1)}[1 + 2\gamma \sum_{m=1}^{N} T_{2N}(x_0 \cos\frac{\theta_m}{2}) \cos m\theta_n],\ \text{where}$$

$$x_0 = \cosh(\tfrac{1}{2N} \cosh^{-1}\gamma),\ \theta_m = \tfrac{2\pi m}{(2N+1)},\ \theta_n = \tfrac{2\pi n}{(2N+1)},$$ and T_{2N} is Chebyshev polynomial. With this window function, we will demonstrate later that a 2h or less cutoff period is sufficient to effectively remove unbalanced high frequency oscillations in early model forecasts.

In order to compute the filtered field $\tilde{f}_n$ in (3), one has to integrate the model back and forth to obtain the time series f_{n-k}. For adiabatic digital filtering, we integrate the model backward and forward adiabatically both from the analysis state for a half cutoff period to obtain the needed f_{n-k} for filtering. For diabatic digital filtering, one can either (a) integrate the model backward adiabatically for a half cutoff period and forward diabatically for a full cutoff period, or (b) integrate the model backward adiabatically for a full cutoff period, apply filtering, and then integrate forward diabatically for a full cutoff period to obtain f_{n-k} (Fig. 2). From the results of many numerical experiments, we have concluded that the two integration procedures for diabatic digital filtering give very similar filtering effects. We, therefore, choose the first integration procedure of the diabatic digital filtering to save cost.

There are several practical decisions we have to make when we apply the diabatic digital filtering to balance initial conditions for mesoscale models. We have to choose a cutoff period, a way to handle boundary conditions during back and forth initialization integrations, a way to handle moving grids, and amounts of numerical diffusion to control numerical noise in space,. The most critical decision is the choice of cutoff period. It not only determines the cost of the dynamic initialization, but also affects the other decisions. For tropical cyclone initialization, the most unbalanced components are of very high frequency, with oscillation periods shorter than a few time steps. Therefore, a relatively short cutoff period should be sufficient to remove the major portion of the unbalanced components. A short cutoff period allows us to choose fixed boundary conditions and fixed moving grids during the initialization integrations. Furthermore, a short cutoff period can prevent very large wind speeds in the backward adiabatic integration that may otherwise exceed the Courant-Freidrichs-Levy (CFL) condition, resulting in instability. We have chosen a 2h or less cutoff period with fixed boundary conditions and fixed moving grids when we implement the DDF initialization to HWRF and COAMPS.

3. Implementation to HWRF and COAMPS

Prior to the start of this project, the dynamic initialization procedure in COAMPS consisted only of an adiabatic digital filter initialization with Lanczos window. In this project, we have added options in COAMPS to choose different windows (Lanczos, Kaiser, or Dolph-Chebyshev) and different integration strategies (adiabatic filtering, diabatic filtering, or diabatic double filtering). We then used COAMPS as our tool to perform numerical experiments in testing the impacts of difference choices, such as the cutoff period, boundary condition treatments, window selection, and integration strategy on COAMPS tropical cyclone forecasts.

Implementing DDF initialization to HWRF is much more difficult than the task for COAMPS. One has to follow specific infrastructure rules set up by Weather Research Forecasting (WRF) model committees. WRF uses Earth System Modeling Framework (ESMF) clock utilities to control time integration and recursive calls to handle do-loops and grid nesting. The code is not all explicitly traceable to numerical modelers. Furthermore, WRF uses a data dictionary (Registry file) to construct a large portion of its source code during the compiling time. This makes code modification a bit more difficult since the changes made to the code may be lost after recompiling. After discussing with HWRF team members and going through several iterations in design, we have developed a DDF package for HWRF that needs only 1 line change to original HWRF source codes, 1 line change to the Registry file, and few line changes in Makefiles for adding new routines.

The package includes 4 “*.F” files, “ddf_init.F”, “ddf_interface.F”, “module_ddf_filter.F” and “module_ddf_integrate.F”.

The routine “ddf_init” is the driver of the DDF initialization that is inserted into the original HWRF routine “wrf_run” right before the “call integrate (head_grid)”. It performs the following functions, (a) allocates work arrays for DDF integrations, (b) allocates all nest grids (similar to codes in routine “integrate”, (c) saves original physics parameters and clock settings to local arrays, (d) computes filter weights, (e) changes clock settings for backward and forward integrations of the dynamic initialization, (f) perform time integration and filtering by calling "ddf_integrate", (g) puts back original physics parameters and clock settings after the initialization has been completed, (h) and deallocates all DDF work arrays. The routine “ddf_interface” is modified from the original HWRF routine “sove_interface” by adding the “call ddfilter” to perform the summation for the inverse Fourier transform in equation (3). The module “module_ddf_filter” contains all the digital filtering routines that compute weights for filtering and the summation. The module “module_ddf_integrate” is modified from the original HWRF module “module_intergrate” by taking out allocation for nest grids (already included in ddf_init) and adding controls to perform adiabatic backward and diabatic forward DDF integration. The new “ddf_interface.F” file is added to the HWRF directory “./share”, and the rest of three new files “ddf_init.F”, “module_ddf_filter.F” and “module_ddf_integrate.f” are added to the HWRF directory “./frame”. With modifications to corresponding Makefiles for new files, inserting “call ddf_init” in “wrf_run” routine, and adding “cutoff_hr” to the Registry_nmm file, the HWRF “compile” script successfully completes the compilation and generates 1 “.a” file and 5 “.exe” files in “./main”.

4. Test and Evaluation

Since the HWRF was also under development with frequent updates during the same time period that this project was conducted, our testing and evaluating DDF initialization for tropical cyclone forecasts are mainly performed with COAMPS. With the choice of a 2h cutoff period, the initial oscillations shown in Fig.1 are greatly reduced after the DDF initialization. (Fig.3). The example shown in Fig.1a and Fig. 1b may be an extreme case since the quality of the initial analysis is very poor that analyzed sea-level pressure and 850 mb wind centers are dislocated (Fig.4). The DDF initialization corrects this dislocation problem (Fig. 5). The sudden shrink of sea-level pressure pattern in early forecast (Figs. 1c, d) comes from another type of unbalance that pressure gradient is not large enough to support the balance with strong wind (Fig 6). As a consequence, the sea-level pressure pattern is quickly adjusted to a tighter pattern in early forecast periods. With a better initial analysis, the initial oscillations for the same model forecast are less serious (Fig.7). However, without any initialization, the initial oscillations are still too large to prohibit forecasters from using the predicted initial tendencies for guidance. With DDF initialization (red lines in Figs. 7c, d), much clear tendencies of a delaying cyclone can be seen from the model forecast. From Fig. 3 and Fig. 7, we can find that the two types of diabatic digital filtering integrations (diab-1 and diab-2) give very similar filtering results, while adiabatic filtering (blue lines) leaves significant parts of unbalanced components in initial conditions after the filtering. Fig. 8 shows another example of the comparison between the 3 types of filtering. Unbalanced initial oscillations are again quite large with adiabatic digital filtering and the two types of diabatic digital filtering produces similar results. Fig. 9 shows a comparison with and without DDF initialization for a tropical cyclone intensifying in early forecast and Fig. 10 shows a similar comparison for an initial decaying case. These examples clearly demonstrate that DDF initialization can improve the quality of tropical cyclone initial tendency forecasts predicted by the model.

The overall impacts of DDF initialization on tropical cyclone track and intensity forecasts should depend strongly on the quality of initial analysis and characters of the dynamic model. For COAMPS, with 58 model forecasts, we find DDF initialization gives small positive impacts on both track and intensity forecasts (Fig.11). The improvement to the track forecast is quite minimal until 48h forecast. The improvement to the intensity forecast is more significant (about 10%) starting from 6h forecast. The degradation of cyclone intensity at initial time may be misleading since the analyzed strong wind may not be maintained by the dynamic model, either due to inadequate resolution and physics or large amounts of unbalanced components from the analysis.

The implementation of DDF initialization to HWRF is tested by constructing an idealized environment that uses prescribed initial conditions and prescribed tendencies for prognostic variables without going through an actual HWRF forecast. The idea is to test the DDF integration flow and the filtering process without interference from other parts of the HWRF forecast. Since the DDF routines all use HWRF routines to get initial conditions and calculate the time changes of prognostic variables, the DDF initialization should work well in HWRF with real data, if it passes this idealized test. A supplemental routine "setinit.f" was written to set up ESMF clock parameters for DDF integrations, assign proper input values to "mod_config_rec" for grid configuration, and prescribe initial values to the prognostic variables. A small driver "testdriver.f" was written to call "setinit" and "ddf_init" for testing the DDF integration. The HWRF routine "solve_nmm" was trimmed to only specify the prognostic variable tendencies. All needed HWRF routines referenced in the "USE" statements of the DDF routines are collected in a working directory to create a library for testing. Some modifications to those routines were made to avoid interaction with the HWRF Input/Output and moving grid setup. After a long debugging process, the testing program successfully runs, and products a printout that verifies the DDF time integrations and filtering process are correctly performed in the HWRF structure (see Appendix 1).

5. Discussion

DDF initialization is designed to remove unbalanced high-frequency oscillations from initial conditions. Therefore, its impacts on model tropical cyclone forecasts depend upon the degree of imbalance in the initial conditions. Initialization for tropical cyclone forecasts may be fundamentally different from initialization for general NWP applications. For synoptic or mesoscale NWP forecasts, initial conditions usually contain a relatively small amount of unbalanced components. Therefore, the NWP forecasts with and without initialization tend to merge together after the initial adjustment oscillations. On the other hand, due to poor quality background fields and an inadequate analysis method, initial conditions for tropical cyclone forecasts usually contain a large amount of unbalanced components. Therefore, the tropical cyclone forecasts with and without initialization may not merge together after the initial oscillations. In fact, significant changes to tropical cyclone track and intensity forecasts may occur in some cases after the DDF initialization.

However, with DDF initialization, forecasters can be confident that the initial intensity and structure tendencies predicted by the model are from the model dynamics rather than unbalanced oscillations. This will add new value to dynamic models for providing forecasters with guidance in tropical cyclone intensity and structure forecasts. In evaluating COAMPS intensity forecasts, we have realized that, after DDF initialization, the predicted initial tendency of central sea level pressure is much more reliable than the predicted initial tendency of maximum wind extracted from the lowest model level. The maximum wind tends to be underestimated after DDF initialization if the tropical cyclone is a strong one initially. In such cases, the maximum wind tends to increase for a few hours after the DDF initialization no matter if the central pressure is dropping or filling. The underestimate in surface wind is interpreted as a result of too large a surface stress applied to the lowest level when very strong wind appears at the lowest level after the frictionless backward integration. The situation is improved by a shorter cutoff period that reduces the wind speed at the lowest level with the shorter backward integration. The shorter cutoff period for those strong cyclone cases has very little impact on their track forecasts.

Appendix 1

List here is a trimmed sample of printout from a test run for DDF implementation to HWRF. The full printout can be generated by running the test script ./testddf/run/run-ddf.cc, included in the delivered tar file.

```
in test - before call setinit
 inside setinit:  #alarm=  5
 inside setinit: done model_config_rec assignment
setinit:  before calling alloc_and_conf
DYNAMICS OPTION: nmm dyncore
 alloc_space_field: domain  1    0
 inside setinit: head_grid%id = 1
 head_grid%alarms(1) is off
 head_grid%alarms(1)%RingTimeSet=  T
 head_grid%alarms(1)%RingIntervalSet= F
 head_grid%alarms(1)%Enabled        = T
 inside setinit: after initializing prognostic variables
 inside setinit: id=  1
 inside setinit: m,n,kk,nbdy=  101 99 18 5
 in test - before call ddf-init, after grid allocation: id =  1
 in test - m,n,kk =  101 99 18 5
 in ddf-init: 0, just in ddf_init, head_grid%id =  1
nesting time
nest_start%YR 2007 |  nest_stop%YR 2007
nest_start%MM 1 |  nest_stop%MM 1
nest_start%DD 15 |  nest_stop%DD 15

parenting time
head_grid%current_time%YR 2007
head_grid%current_time%MM 1
head_grid%current_time%DD 15
DYNAMICS OPTION: nmm dyncore
 alloc_space_field: domain  2    0
 from med_nest_initial: initialize inner nest OK, id=  2
 in ddf-init: 1, found child mesh, id,dt=  2 111 0 1 0 0 0 F 0 0
      0
 in ddf-init: 2, ncount, imore =  2 1
nesting time
nest_start%YR 2007 |  nest_stop%YR 2007
nest_start%MM 1 |  nest_stop%MM 1
nest_start%DD 15 |  nest_stop%DD 15
parenting time
head_grid%current_time%YR 2007
head_grid%current_time%MM 1
head_grid%current_time%DD 15
DYNAMICS OPTION: nmm dyncore
 alloc_space_field: domain  3    0
 from med_nest_initial: initialize inner nest OK, id=  3
 in ddf-init: 3, found deeper child mesh, id,dt =  3 37 0 1 0 0 0
      F 0 0 0 3
 in ddf-init: 4, for deeper child mesh, id =  3 1
 in ddf-init: 4, for deeper child mesh, id =  3 0
 in ddf-init: 5, mmmx =  144
 in ddf-init: 6, nest,ims,ime,jms,jme,kms,kme=  1 1 101 1 99
      1 18
 in ddf-init: 6, nest,ims,ime,jms,jme,kms,kme=  2 1 78 1 92 1
      18
 in ddf-init: 6, nest,ims,ime,jms,jme,kms,kme=  3 1 65 1 91 1
      18
hn0-1 =  0.6250000000E-01 1
hnsum =  0.9800289273 1 16
hn0-1 =  0.2083333395E-01 2
hnsum =  0.9794594049 2 48
hn0-1 =  0.6944444496E-02 3
hnsum =  0.9796720147 3 144


---
---
inside ddf_integrate- 1: stop_subtime for, id=  1 101 18 99
inside ddf_integrate- 2: inside DO-WHILE:=  0 1
inside solve_nmm: id,dt,istep =  1 -333.0000000 0
inside solve_nmm: pd,u,v,t =  997.2249756 7.224999905 -
     1.775000095 284.4500122
inside solve_nmm: pd,u,v,t =  997.2249756 7.224999905 -
     1.775000095 284.4500122
inside ddf_face: imode =  1
inside ddf_integrate- 3: inside child-DO WHILE:=  1 1
 inside ddf_integrate- 4: before ddf_integrate=  2 111 0 1 0 0
      0 F 0 0 0
 inside ddf_integrate- 1: stop_subtime for, id=  2 78 18 92
 inside ddf_integrate- 2: inside DO-WHILE:=  0 2
 inside solve_nmm: id,dt,istep =  2 -111.0000000 0
 inside solve_nmm: pd,u,v,t =  1029.074951 19.07500076
      1.075000048 308.1499939
 inside solve_nmm: pd,u,v,t =  1029.074951 19.07500076
      1.075000048 308.1499939
 inside ddf_face: imode =  1
 inside ddf_integrate- 3: inside child-DO WHILE:=  1 2
 inside ddf_integrate- 4: before ddf_integrate=  3 37 0 1 0 0 0
      F 0 0 0
 inside ddf_integrate- 1: stop_subtime for, id=  3 65 18 91
 inside ddf_integrate- 2: inside DO-WHILE:=  0 3
 inside solve_nmm: id,dt,istep =  3 -37.00000000 0
 inside solve_nmm: pd,u,v,t =  1044.691650 29.69166756
      2.691666603 319.3833313
 inside solve_nmm: pd,u,v,t =  1044.691650 29.69166756
      2.691666603 319.3833313
 inside ddf_face: imode =  1
 inside ddf_integrate- 3: inside child-DO WHILE:=  1 3
 inside ddf_integrate- 5: after  DO WHILE  3
 inside ddf_integrate- 2: inside DO-WHILE:=  1 3
 inside solve_nmm: id,dt,istep =  3 -37.00000000 1
 inside solve_nmm: pd,u,v,t =  1044.383301 29.38333511
      2.383333206 318.7666626
 inside solve_nmm: pd,u,v,t =  1044.383301 29.38333511
      2.383333206 318.7666626
 inside ddf_face: imode =  1
 inside ddf_integrate- 3: inside child-DO WHILE:=  2 3
 inside ddf_integrate- 5: after  DO WHILE  3
 inside ddf_integrate- 2: inside DO-WHILE:=  2 3
 inside solve_nmm: id,dt,istep =  3 -37.00000000 2
 inside solve_nmm: pd,u,v,t =  1044.074951 29.07500267
      2.074999809 318.1499939
 inside solve_nmm: pd,u,v,t =  1044.074951 29.07500267
      2.074999809 318.1499939
 inside ddf_face: imode =  1
 inside ddf_integrate- 3: inside child-DO WHILE:=  3 3
 inside ddf_integrate- 5: after  DO WHILE  3
 inside ddf_integrate- 6: finish time intgration
 inside ddf_integrate- 8: finish 1 cycle, imode,istep,id=  1 3 3
 inside ddf_integrate- 5: after  DO WHILE  2
 inside ddf_integrate- 2: inside DO-WHILE:=  1 2
 ---
 ---
 inside ddf_face: imode =  1
 inside ddf_integrate- 3: inside child-DO WHILE:=  143 3
 inside ddf_integrate- 5: after  DO WHILE  3
 inside ddf_integrate- 2: inside DO-WHILE:=  143 3
 inside solve_nmm: id,dt,istep =  3 -37.00000000 143
 inside solve_nmm: pd,u,v,t =  1026.499023 11.50007915 -
      15.50000286 282.9998779
 inside solve_nmm: pd,u,v,t =  1026.499023 11.50007915 -
      15.50000286 282.9998779
 inside ddf_face: imode =  1
 inside ddf_integrate- 3: inside child-DO WHILE:=  144 3
 inside ddf_integrate- 5: after  DO WHILE  3
 inside ddf_integrate- 6: finish time intgration
 inside ddf_integrate- 7a: imode=1 after initialize ddf arrays=
      id, 3 144
 inside ddf_integrate- 8: finish 1 cycle, imode,istep,id=  1 144
      3
 inside ddf_integrate- 5: after  DO WHILE  2
 inside ddf_integrate- 6: finish time intgration
 inside ddf_integrate- 7a: imode=1 after initialize ddf arrays=
      id, 2 48
 inside ddf_integrate- 8: finish 1 cycle, imode,istep,id=  1 48 2
 inside ddf_integrate- 5: after  DO WHILE  1
 inside ddf_integrate- 6: finish time intgration
 inside ddf_integrate- 7a: imode=1 after initialize ddf arrays=
      id, 1 16
 inside ddf_integrate- 8: finish 1 cycle, imode,istep,id=  1 16 1
 in ddf-init: 10, after call to ddf_integrate, imode=  1
 inside child_time -1, kid, id =  1 2
 inside child_time -2, kid, id, ncount =  1 2 2
 id, timestep =  2 111 0 1 0 0 0 F 0 0 0
 inside child_time -3, more to search, ncc, gridid =  2 2
 inside child_time -4 ,       kid, id, ncount =  1 3 3
 id, timestep =  3 37 0 1 0 0 0 F 0 0 0
 inside child_time -5 , imore, ncount, ncc=  1 3 2
 inside child_time -3, more to search, ncc, gridid =  3 3
 inside child_time -5 , imore, ncount, ncc=  0 3 3
 in ddf-init: 9b, after call to child_time for mode=2
 inside ddf_integrate- 1: stop_subtime for, id=  1 101 18 99
 inside ddf_integrate- 2: inside DO-WHILE:=  0 1
 inside solve_nmm: id,dt,istep =  1 333.0000000 0
 inside solve_nmm: pd,u,v,t =  983.3496704 -6.650001049 -
      15.64999962 256.7001343
 inside solve_nmm: pd,u,v,t =  983.3496704 -6.650001049 -
      15.64999962 256.7001343
 in ddfilter nx,istep,mx,hnw,:  1 0 16 0.1300427946E-02
 in ddfilter: ddfv-p,u,t:  1 0 1.278775334 -0.8647846989E-02
      0.3338200152
 inside ddf_face: id, nx,istep =  1 1 0
 inside ddf_face: imode =  2
 inside ddf_integrate- 3: inside child-DO WHILE:=  1 1
 inside ddf_integrate- 4: before ddf_integrate=  2 111 0 1 0 0
      0 F 0 0 0
 inside ddf_integrate- 1: stop_subtime for, id=  2 78 18 92
 inside ddf_integrate- 2: inside DO-WHILE:=  0 2
 inside solve_nmm: id,dt,istep =  2 111.0000000 0
 inside solve_nmm: pd,u,v,t =  1011.499634 1.500002384 -
      16.50000191 272.9998779
 inside solve_nmm: pd,u,v,t =  1011.499634 1.500002384 -
      16.50000191 272.9998779
 in ddfilter nx,istep,mx,hnw,:  2 0 48 0.1215211232E-03
 in ddfilter: ddfv-p,u,t:  2 0 0.1229185686 0.1822819759E-03
      0.3317525238E-01
 inside ddf_face: id, nx,istep =  2 2 0
 inside ddf_face: imode =  2
 inside ddf_integrate- 3: inside child-DO WHILE:=  1 2
 inside ddf_integrate- 4: before ddf_integrate=  3 37 0 1 0 0 0
      F 0 0 0
 inside ddf_integrate- 1: stop_subtime for, id=  3 65 18 91
 inside ddf_integrate- 2: inside DO-WHILE:=  0 3
 inside solve_nmm: id,dt,istep =  3 37.00000000 0
 inside solve_nmm: pd,u,v,t =  1026.807373 11.80841255 -
      15.19166946 283.6165466
 inside solve_nmm: pd,u,v,t =  1026.807373 11.80841255 -
      15.19166946 283.6165466
 in ddfilter nx,istep,mx,hnw,:  3 0 144 0.1273696034E-04
 in ddfilter: ddfv-p,u,t:  3 0 0.1307840459E-01
      0.1504032844E-03 0.3612412605E-02
 inside ddf_face: id, nx,istep =  3 3 0
 ---
 ---
 inside ddf_face: imode =  2
 inside ddf_integrate- 3: inside child-DO WHILE:=  32 1
 inside ddf_integrate- 4: before ddf_integrate=  2 111 0 1 0 0
      0 F 0 0 0
 inside ddf_integrate- 1: stop_subtime for, id=  2 78 18 92
 inside ddf_integrate- 2: inside DO-WHILE:=  93 2
 inside solve_nmm: id,dt,istep =  2 111.0000000 93
 inside solve_nmm: pd,u,v,t =  1047.577148 37.57497406
      19.57499886 345.1501160
 inside solve_nmm: pd,u,v,t =  1047.577148 37.57497406
      19.57499886 345.1501160
 in ddfilter nx,istep,mx,hnw,:  2 93 48 0.2647980582E-03
 in ddfilter: ddfv-p,u,t:  2 93 1029.318237 19.44078445
      308.8487549
 inside ddf_face: id, nx,istep =  2 2 93
 inside ddf_face: imode =  2
 inside ddf_integrate- 3: inside child-DO WHILE:=  94 2
 inside ddf_integrate- 4: before ddf_integrate=  3 37 0 1 0 0 0
      F 0 0 0
 inside ddf_integrate- 1: stop_subtime for, id=  3 65 18 91
 inside ddf_integrate- 2: inside DO-WHILE:=  279 3
 inside solve_nmm: id,dt,istep =  3 37.00000000 279
 inside solve_nmm: pd,u,v,t =  1061.034180 46.03345108
      19.03331566 352.0667725
 inside solve_nmm: pd,u,v,t =  1061.034180 46.03345108
      19.03331566 352.0667725
 in ddfilter nx,istep,mx,hnw,:  3 279 144 0.1244540617E-03
 in ddfilter: ddfv-p,u,t:  3 279 1044.203979 29.61127853
      319.1184082
 inside ddf_face: id, nx,istep =  3 3 279
 ---
 ---
 inside ddf_face: imode =  2
 inside ddf_integrate- 3: inside child-DO WHILE:=  287 3
 inside ddf_integrate- 5: after  DO WHILE  3
 inside ddf_integrate- 2: inside DO-WHILE:=  287 3
 inside solve_nmm: id,dt,istep =  3 37.00000000 287
 inside solve_nmm: pd,u,v,t =  1062.575928 47.57512665
      20.57497597 355.1501160
 inside solve_nmm: pd,u,v,t =  1062.575928 47.57512665
      20.57497597 355.1501160
 in ddfilter nx,istep,mx,hnw,:  3 287 144 0.0000000000E+00
 in ddfilter: ddfv-p,u,t:  3 287 1044.629761 29.63002968
      319.2601929
 inside ddf_face: id, nx,istep =  3 3 287
 inside ddf_face: imode =  2
 inside ddf_integrate- 3: inside child-DO WHILE:=  288 3
 inside ddf_integrate- 5: after  DO WHILE  3
 inside ddf_integrate- 6: finish time intgration
 inside ddf_integrate- 7b: imode=2 after initialize ddf arrays=
      id, 3 288
   after -7b: pd,u,v,t =  1044.629761 29.63002968 2.630000353
      319.2601929
 ZEROED OUT PRECIP/RUNOFF ARRAYS
 ZEROED OUT SFC EVAP/FLUX ARRAYS
 ZEROED OUT ACCUMULATED SHORTWAVE FLUX
      ARRAYS
 ZEROED OUT ACCUMULATED LONGWAVE FLUX
      ARRAYS
 ZEROED OUT ACCUMULATED LATENT HEATING
      ARRAYS
   inside ddf_integrate- 7c: imode=2 after reset buckets 1 64 1
      90 1 17
   inside ddf_integrate- 7d: imode=2 before calling qprint:  3 64
      90
 ---
 ---
   inside ddf_integrate- 8: finish 1 cycle, imode,istep,id=  2 32 1
   in ddf-init: 10, after call to ddf_integrate, imode=  2
   in ddf-init: 11, before put back original namelist
   in ddf-init: 12, before put back original clock
   in putback: 0, inside putback
   in putback: 1, after grid1, id=  1
   id, timestep =  1 333 0 1 0 0 0 F 0 0 0
   in putback: 2, in do-kid, kid, id =  1 2
   id, timestep =  2 111 0 1 0 0 0 F 0 0 0
   in putback: 3, ncount =  2
   in putback: 4, ncc =  2
   in putback: 5, do-kid2  1
   id, timestep =  3 37 0 1 0 0 0 F 0 0 0
   in putback: 6, before sibling
   in putback: 7, imore =  1 3 2
   in putback: 4, ncc =  3
   in putback: 6, before sibling
   in putback: 7, imore =  0 3 3
   in ddf-init: 13, after set back original clock
   good stop in driver
 Job  /opt/lsf/bin/poejob -euilib us ../test.exe
```

Reference

Huang, X.-Y., and P. Lynch, 1993: Diabatic digital-filtering initialization: Application to the HIRLAM model. Mon. Wea. Rev., **121**, 589-603.

Lynch, P., and X.-Y. Huang, 1992: Initialization of the HIRLAM model using a digital filter. Mon. Wea. Rev., **120**, 1019-1034

Rasch, P. J., 1985: Developments in normal model initialization. Part 1: A simple interpretation for normal model initialization. Mon. Wea. Rev., **113**, 1746-1752.

Williamson, D. L., and C. Temperton, 1981: Normal mode initialization for a multi-level gridpoint model. Part 2: Nonlinear aspects. Mon. Wea. Rev., **109**, 744-757.

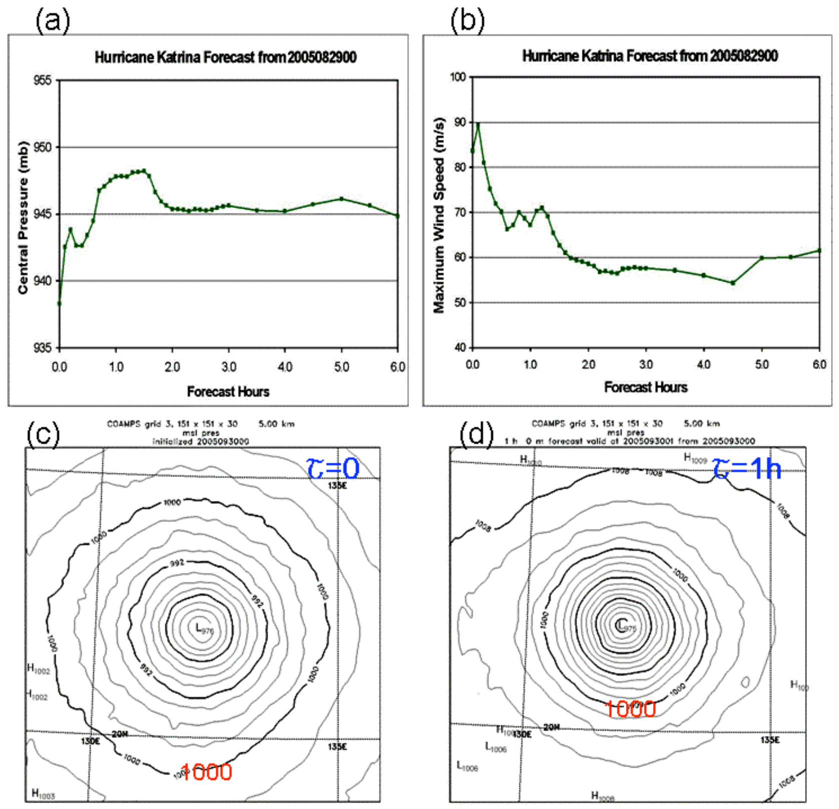

Fig. 1, Example of large initial oscillations by unbalanced initial conditions: first 6h forecast of (a) central sea-level pressure and (b) maximum wind speed for Katrina forecast from 0000UTC 29 August 2005, (c) sea-level pressure analysis and (d) 1h forecast of sea-level pressure for Katrina forecast from 0000UTC 30 August 2005.

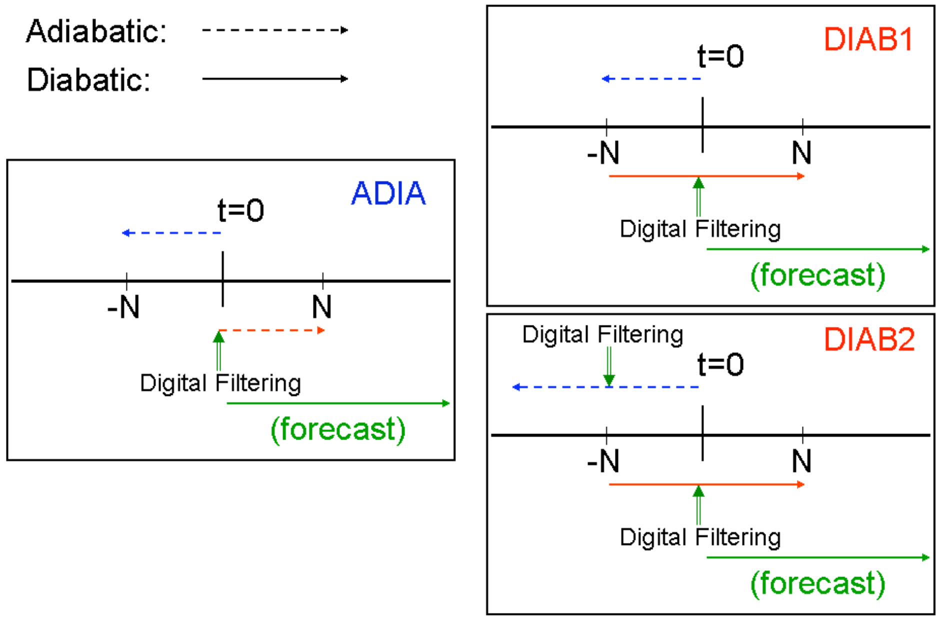

Fig.2. Schematic diagrams of time integration procedures for adiabatic and diabatic digital filter initialization.

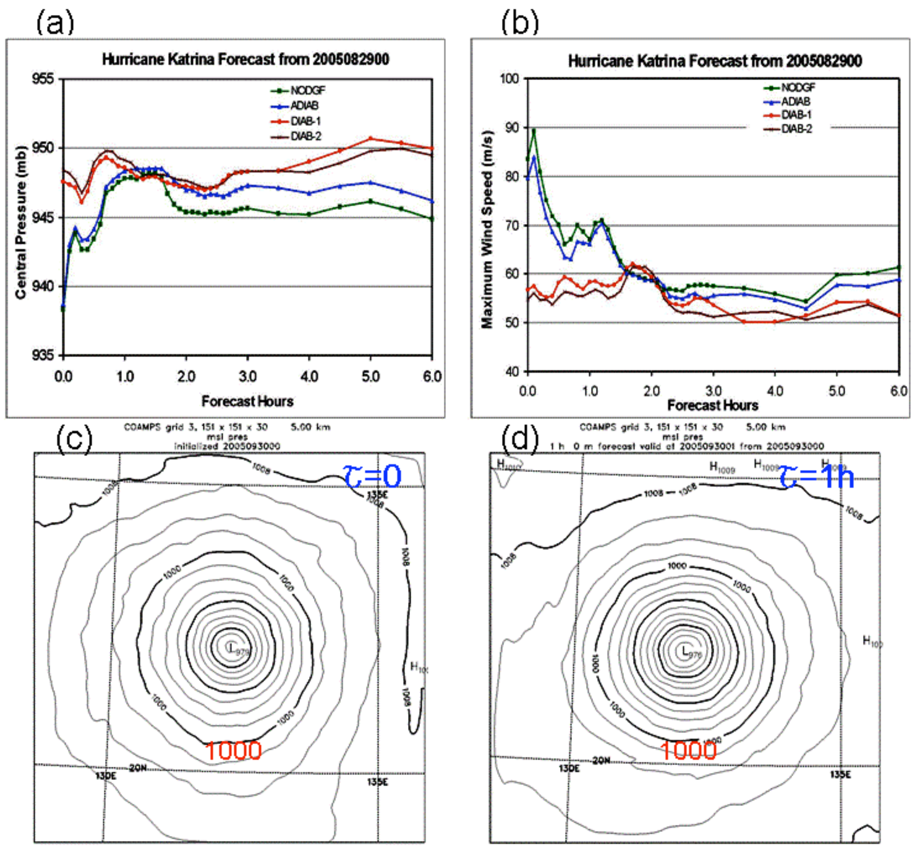

Fig. 3. Same as Fig.1, except after diabatic digital filter initialization (red). The blue lines are for adiabatic and brown lines are for type-2 diabatic filtering (DIAB2).

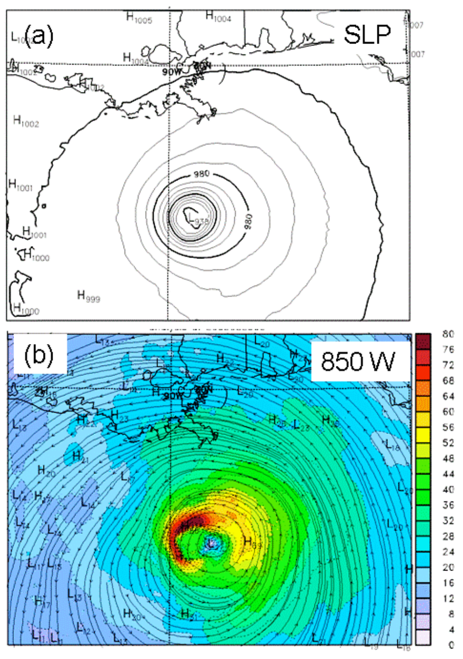

Fig. 4. Initial analysis used in Fig. 1, (a) sea-level pressure and (b) 850 mb wind.

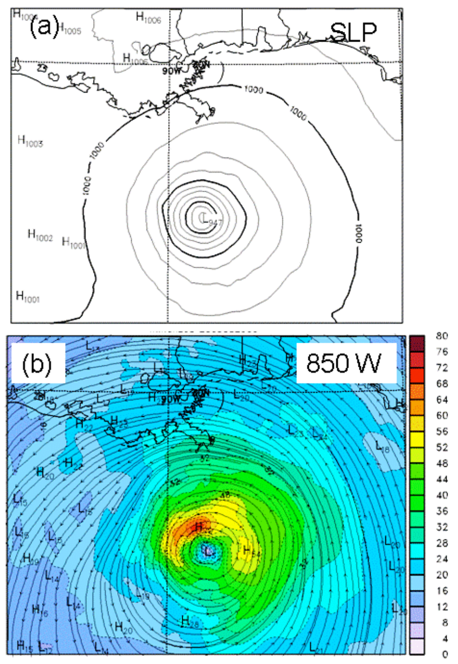

Fig. 5. Same as Fig. 4, except after DDF Initialization.

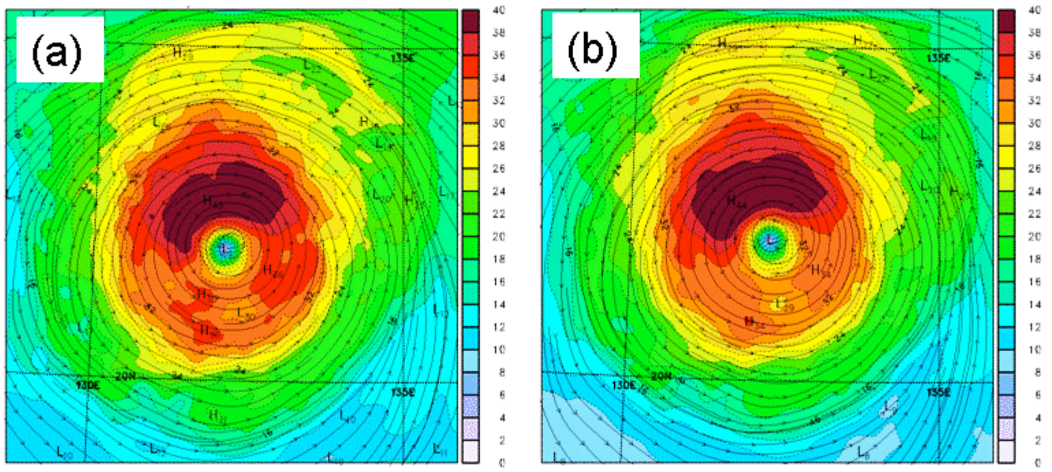

Fig. 6. (a) analyzed and (b) DDF initialized 850 mb wind at 0000UTC 30 August 2005 used for Katrina forecasts shown in Fig. 1c and Fig. 3c.

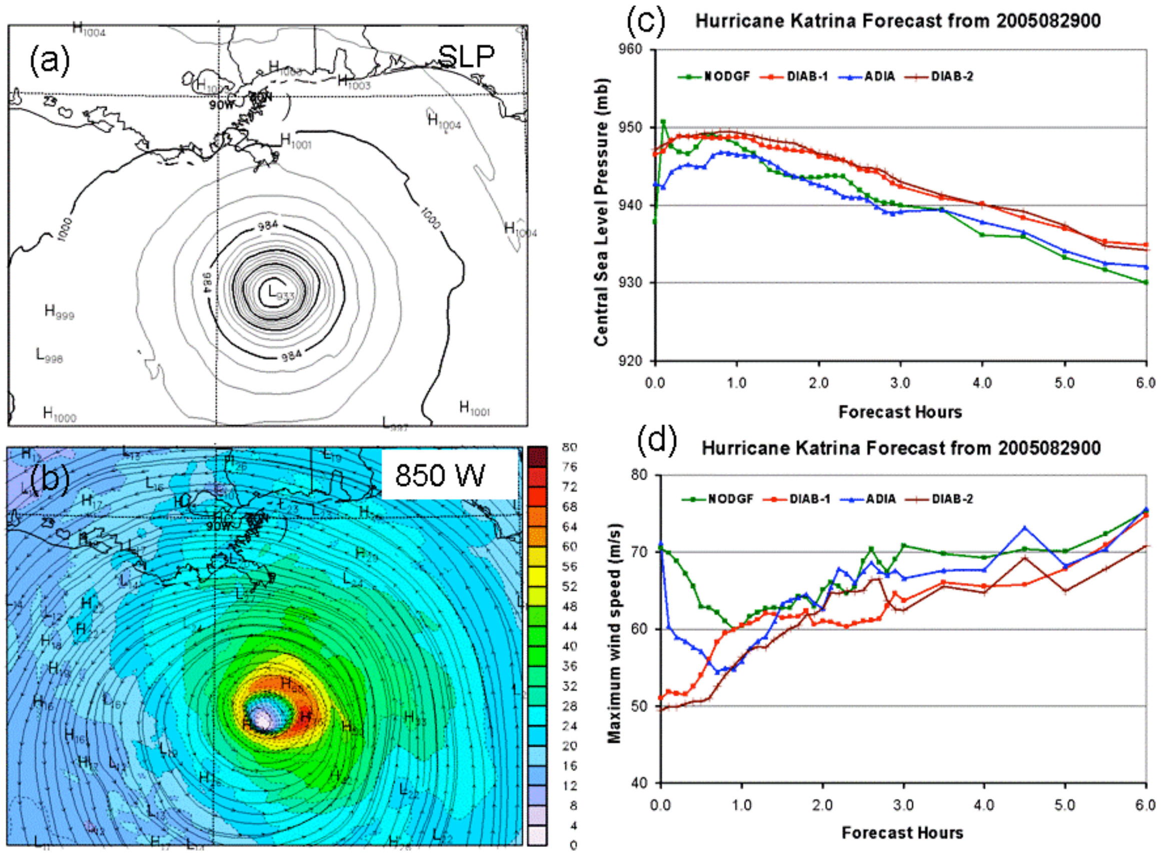

Fig. 7. Another Katrina forecast from 0000UTC 29 August 2005 with a better analysis (a) sea-level analysis, (b) 850mb wind analysis, and first 6h forecast of (c) central sea-level pressure, and (d) maximum wind speed without initialization (green), with adiabatic (blue), diabatic (red), or diabatic type-2 initialization.

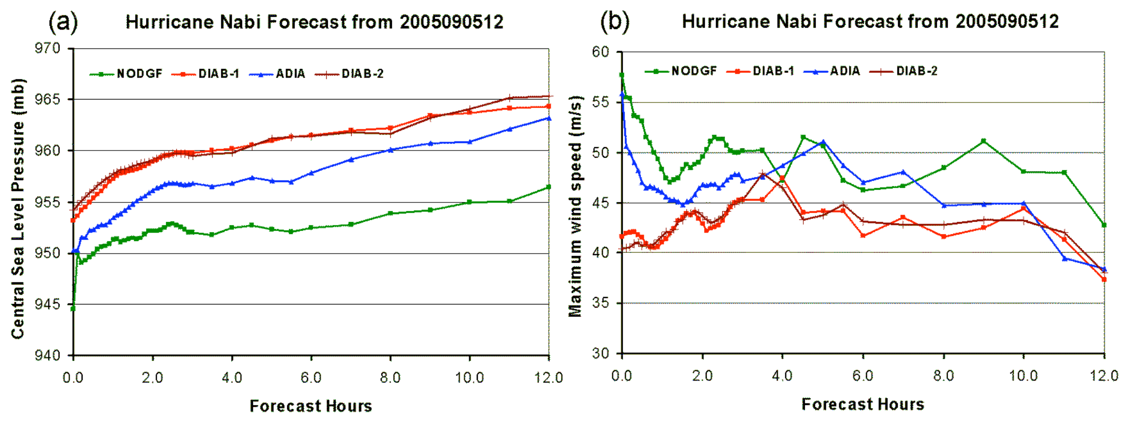

Fig. 8. Same as Fig. 7c and Fig. 7d, except for tropical cyclone Nabi forecast starting from 1200 UTC 5 September 2005.

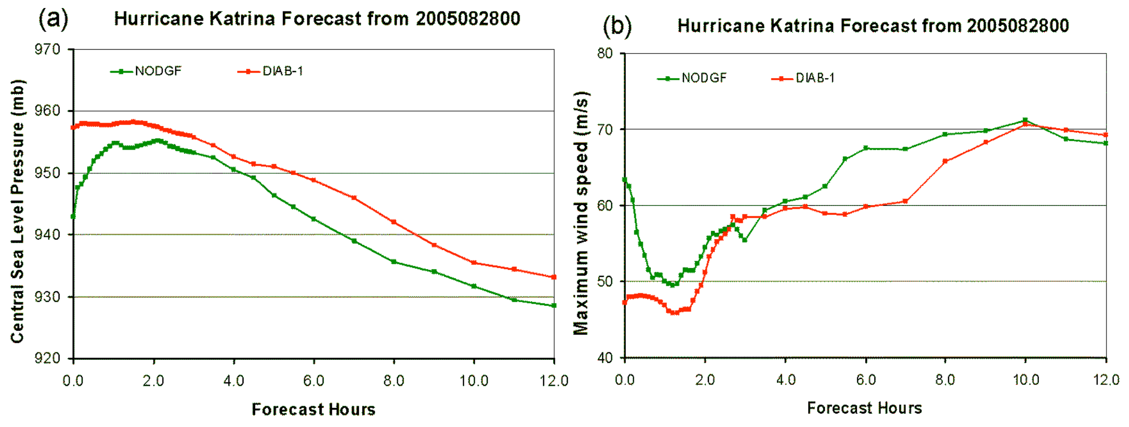

Fig. 9. Forecasts for Katrina from 0000UTC 28 August 2005 with (red) and without DDF initialization, (a) central sea-level pressure and (b) maximum wind speed.

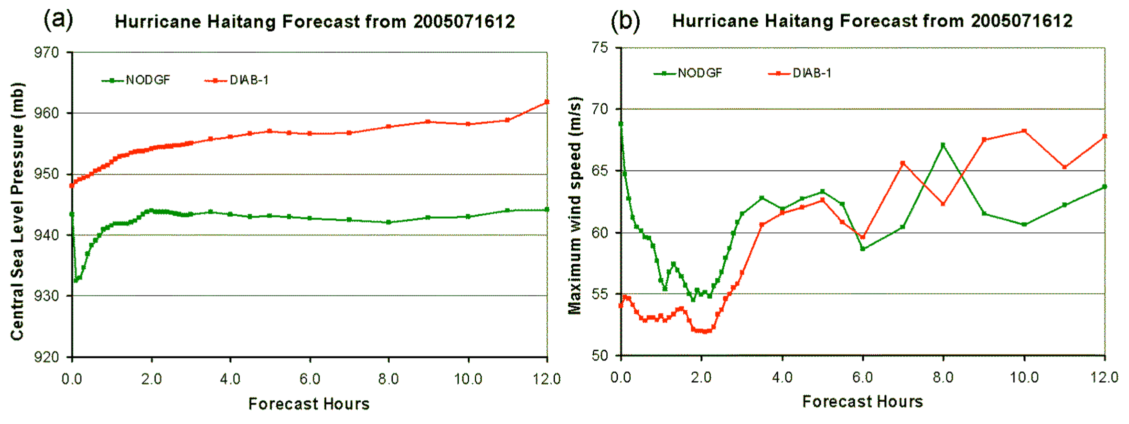

Fig. 10. Same as Fig. 9, except for Haitang from 1200UTC 16 July 2005.

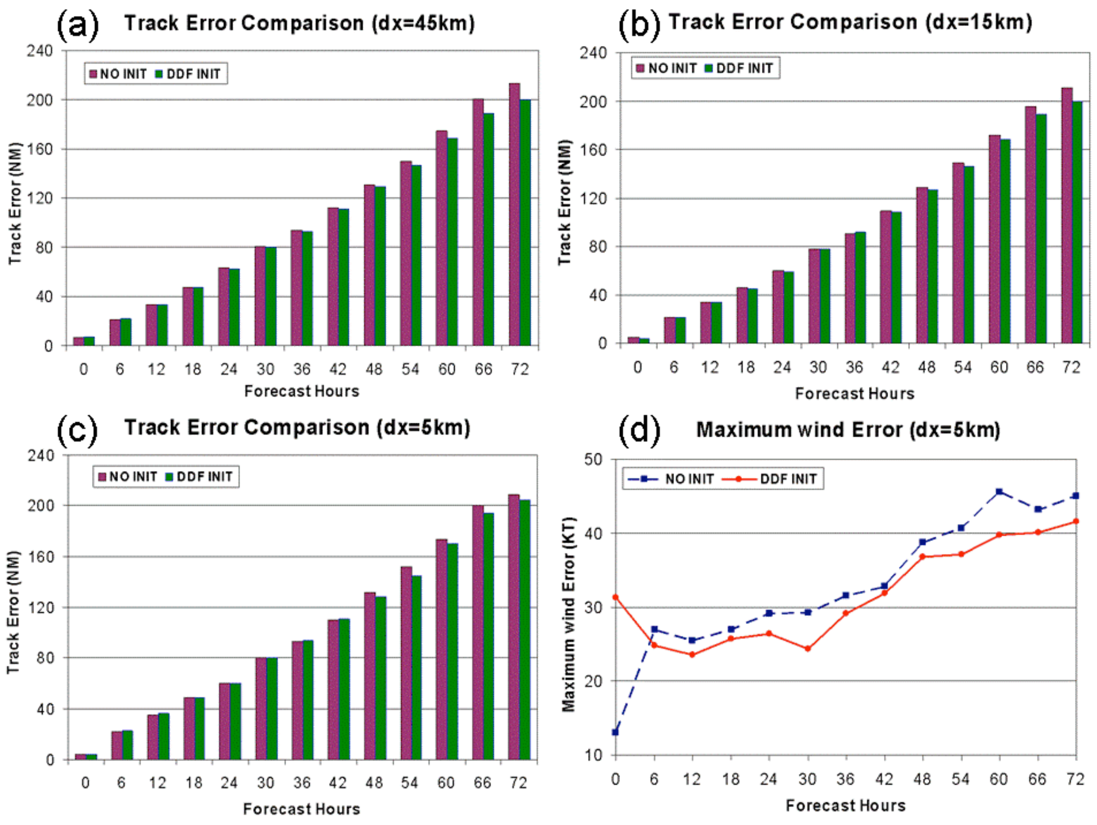

Fig. 11. Comparisons of COAMPS 58 forecast statistics with (green bars or red line) and without (purple bars or blue line) DDF initialization, (a) track error of 45-km mesh, (b) track error of 15-km mesh, (c) track error of 5-km mesh, and (d) maximum wind speed error of 5-km mesh. Observed cyclone positions and maximum wind speeds received from TPC or JTWC are used in computing the forecast errors.